Welcome to Programming

Introduction

Computer programming is by far the biggest employment area in the world of computing.

A programmer will receive a handsome salary if they fulfill all of their objectives. However, the job of a computer programmer can be a stressful one. They will may have to work long hours, especially if there is a deadline looming. They must also have excellent team working and problem solving skills.

Another skill that the programmer must have is the ability to sit down at a computer and think through a problem in a logical step by step fashion. If there are errors in his/her program, the programmer must have the patience to carefully look through the program in order to discover the error that they have made.

Remember:

- The interpreter translator program will not be able to translate the high-level language into binary if there are any errors present in the program code.

- The computer will always produce the correct answer but only if it has been programmed correctly by the human. This means that a program may still work correctly even if it has errors. Therefore a program MUST be tested!

During these tasks, you will create 10 programs. It is vital that you understand what is going on in each program. Your teacher will go through each program with you at the beginning of the lesson. If you’re stuck, ask questions, but do try to fix any programming errors on your own.

These tasks are designed to help you do well in the final two programming assessment tasks. You will even be able to copy and paste sections of program code in order to help you complete these coursework exercises.

Good Luck!

LiveCode

LiveCode is a modern programming environment that was created by a company called RunRev (www.runrev.com). Users can use LiveCode to create any type of program. This could range from a simple application which performs addition to a more advanced game application that could be run on a desktop computer or mobile phone.

It is an event-driven programming language which basically means that it involves the triggering of events such as a mouse click on a button or text entry into an output field.

The LiveCode programming environment has already been installed and can be found in:

![Applications] [LiveCode]

Before you start using LiveCode, you need to manually activate it on your machine and logged in under your account using the passphrase of macmini.

Your teacher will show you how to do this.
Getting Started

Task 1A: Display My Message (your first ever program!)

Specification

A simple program required to ask for and display a simple message in an output box.

The user should select the “Display My Message” button were they will be prompted for their message five times.

In order to display the message five times, a simple \texttt{REPEAT...END REPEAT} loop will need to be added.

Analysis

A programmer always begins by writing out an analysis of the problem. This is when they come up with the steps which need to be carried out by the program in order to sol.

Here is the analysis for the Display My Message Program:

I have been asked to create a simple program which displays a message on the screen 5 times.

I will do this task by first displaying a message on the screen asking for a message to be entered. The user will then key in their message and when they select ok, their message will be displayed in an output box four times using a repeat...until loop.

Design

After the analysis comes the design of the program. The design goes into more detail with regards to the main steps of the program.

The design of a program is also known as the Pseudocode (pronounced sue-do-code). Pseudocode is a cross between our language and programming language.

Here is the design showing steps for the Display My Message Program:

\textbf{Stepwise Design (the main steps of the program)}

1. Set up Variables (areas) to store the users answer
2. Ask for the users message
3. Display the Message

\textbf{Stepwise Refinement (breaking down the main steps into smaller steps)}

1. Set up Variables (areas) to store the users answer
   1.1 set up the variable called my\_message
2. Ask for the users message
   2.1 ask for the users message
   2.2 user puts in their message and this is stored in my\_message
3. Display the Message
   3.1 start a repeat with loop 1 to 5
   3.2 display my\_message into the field called output
   3.3 end repeat
Getting Started

Task 1A: Display My Message

Implementation

The implementation involves going onto the computer and actually creating the program using the analysis and design sections to help.

For the purpose of these exercises, the program code will be supplied for you. However, you will need to create the program code from scratch when you come onto doing your SQA coursework.

Open the “Display My Message” program template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 1A...Display My Message.livecode

Once the program has been opened, select the ‘edit tool’ to edit the script for the “Display My Message” program.

Select the “Display My Message” button to edit the object’s script.

Select the ‘Edit Script’ button at the top on the toolbar.

You will now enter the new lines on the next page carefully.

These new lines will add a loop in order to display your message 5 times.
Task 1A: Display My Message

Implementation (continued)

When you edit the “Display My Message” button, you see the program script that is assigned to the button. This is the action that will be performed when the button is pressed using the mouse.

Your task is to add the program code shown below carefully. Watch your spelling as all your code will need to be correct in order for your program to run successfully!

```livecode
on mouseUp
  // This event lists the components that will occur when the button for 'Display
  // My Message' has been pressed.
  get_message
  display_message
end mouseUp

  // This is a list of variables used in the Display My Message Program
global my_message

on get_message
  // Ask the user for the simple message they wish to be displayed
  ask "Please print your simple message, i.e. Hello World"
  put it into my_message
end get_message

on display_message
  // Display the users message on the screen 5 times in a loop
  repeat with loop = 1 to 5
    put my_message & return after field "output"
  end repeat
end display_message
```

Testing

Test that your program produces the correct output by running it. If the program works correctly, it will display your message five times.

To run a RevMedia program, save and close the script and exit edit mode by selecting the following button:

Select the “Display My Message” button and a pop-up window will appear. From there, key in the message you would like displayed.

This message should appear five times in the output box as shown on the right.
Task 1B: Conversation

Specification

A simple program has been developed to ask the user a couple of questions. This includes the user’s name and favourite subject.

Your task is to open the program and change the questions to questions of your choice.

This could include the persons favourite film, football team or colour, etc.

Let's get to know each other!

Press this button to talk to me!

Open the “Conversation Program” template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 1B...Conversation Program.livecode

Implementation

Click on the ‘edit tool’ to edit the script for the “Conversation” program.

Select the ‘Talk To Me’ button to edit the object script.

Now select the ‘Edit Script’ button at the top on the toolbar.

Add another two questions of your choice to the program script.

Save and close your script. Run the program by selecting the following button:

Save your program in your programming folder on the server as Task 1B: Conversation.
Task 2: Party Invitation Program

Specification

A program is required to take in the title, location, time and date of a party and display these details in the form of a party invitation on the screen.

The graphical user interface for the program has already been created for you. All you need to do is to key in the program for the button “Create Invitation”.

Before starting, familiarise yourself with both the analysis and design below.

Analysis

I have been asked to create a program which will get the title, location, time and date of a party and display these details in an output box in the form of a party invitation.

I will do this task by first displaying a message on the screen asking for the user to enter the title of their party. Once the user has entered this, I will then display another message asking for party location and these details will be entered. I will then ask the user a further two questions, these questions will ask for the time and date of the party. I will expect the user to key in these details also.

All details entered will then be displayed in the form of an invitation in a field called “output”.

Design

Here is the design showing the steps for the Party Invitation Program:

Stepwise Design (the main steps of the program):

1. Set up the Variables (areas) to store the invitation details
2. Get Invitation Details
3. Display Invitation Details

Stepwise Refinement (breaking down the main steps into smaller steps):

1. Set up the Variables (areas) to store the invitation details
   1.1 set up the variables called invitation_title, Invitation_Location, Invitation_Time, Invitation_Date
2. Get Invitation Details
   2.1 ask for the title of the user’s party
   2.2 user puts in the title and this is stored in invitation_title
   2.3 ask for the location of the user’s party
   2.4 user puts in the location and this is stored in invitation_location
   2.5 ask for the time of the user’s party
   2.6 user puts in the time and this is stored in invitation_time
   2.7 ask for the date of the user’s party
   2.8 user puts in the date and this is stored in invitation_date
3. Display Invitation Details
   3.1 display the text You’re invited to followed by invitation_title into the field called output
   3.2 display the text The location is followed by invitation_location into the field called output
   3.3 display the text The time is followed by invitation_time into the field called output
   3.4 display the text The date is followed by invitation_date into the field called output
   3.5 display the text Don’t be late into the field called output
   3.6 display a blank line
Task 2: Party Invitation Program

Please Read!

Your teacher will go over this program with you before you begin. Make sure you listen carefully and work out what the code on the next page is causing the program to do.

As you progress through the tasks, they will start to get a little harder. If you run into difficulties, for example, your program does not run, the line the error is on will be highlighted for you. Try to work out for yourself what the problem could be before calling on the help of the teacher.

Implementation

Open the “Party Invitation” template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 2...Party Invitation.livecode
Task 2: Party Invitation Program

Implementation (continued)

Click on the ‘edit tool’ to edit the script for the Invitation program.

Select the “Create Invitation” button to edit the object script using the button on the toolbar.

Your task is to add the program code shown below carefully. Watch your spelling as all your code will need to be correct in order for your program to run successfully!

```
on mouseUp
  // This event lists the components that will occur when the button for 'Create Invitation' has been pressed.
  Get_Invitation_Details
  Display_Invitation_Details
end mouseUp

  // This is a list of variables used in the Party Invitation Program.
global Party_Title, Party_Location, Party_Time, Party_Date

  on Get_Invitation_Details
    // This component will get the party invitation details from the user.
    ask "Please enter the title of your party, for example, Easter Chicks Party"
    put it into Party_Title
    ask "Please enter the location of your party, for example, Electric Circus Karaoke Club, Edinburgh"
    put it into Party_Location
    ask "Please enter the time of your party, for example, 8pm until 1am"
    put it into Party_Time
    ask "Please enter the date of your party, for example, 8th April 2010"
    put it into Party_Date
  end Get_Invitation_Details

  on Display_Invitation_Details
    // This component will display the details of the party invitation in the output field.
    put "You're invited to " & Party_Title & return after field "Output"
    put "The location is " & Party_Location & return after field "Output"
    put "The time is " & Party_Time & return after field "Output"
    put "The date is " & Party_Date & return after field "Output"
    put "Don't be late!!" & return after field "Output"
  end Display_Invitation_Details
```
Task 2: Party Invitation Program

Testing

Once your program is complete, check that it works correctly by clicking on the ‘Compile button’.

Fix any errors in your program. The lines that contain errors will be picked up by the translator for you to fix.

When you have no errors, you are ready to test that your application program works correctly by running it.

Click on the ‘Run Tool’ to run your application program. Click on the ‘Create Invitation’ button and key in any data you wish. Your program should show similar output to the screenshot below.

Party Invitation

You’re invited to Steve’s 30th Birthday Party
The location is The Balmoral Hotel, Edinburgh
The time is 6pm until 1am
The date is 10th September 2011
Don’t be late!!

Task 2 (b): Party Invitation Program - 4 party Tickets using a loop

Go back into the script of the ‘Create Invitation’ button and enter the three new lines shown below. This will set up a fixed loop to produce the party ticket 4 times as well as print the number of the party ticket. Test that your program works by ensuring that it produces four tickets with the ticket number.

on Display_Invitation_Details
   // This component will display the details of the party invitation in the output field
   repeat with ticket = 1 to 4
      put "This is ticket number: " & ticket & return after field "Output"
      put "You’re invited to " & Party_Title & return after field "Output"
      put "The location is " & Party_Location & return after field "Output"
      put "The time of this party is " & Party_Time & return after field "Output"
      put "The date of this party is " & Party_Date & return after field "Output"
      put "Don’t be late!!" & return after field "Output"
      put return after field "Output"
   end repeat
end Display_Invitation_Details

Save your program in your programming folder on the server as Task 2: Party Invitation.
What are Variables?

Let’s talk about variables as they are very important in programming.

To put it simply, a variable is a "box" into which data can be placed. We give these boxes names which suggest or give us a clue as to what data is being held in the box.

Here are the variables that we used in the Party Invitation Program.

**Invitation_Title**
- Sarah’s Sweet Sixteenth

**Invitation_Time**
- 6pm until 12am

**Invitation_Date**
- 12 May 2011

Variables cannot contain any spaces and must not be a reserved command in RevMedia. You can tell if a variable has been accepted as it will appear in black font colour when typed into the script window as shown below:

```
put "You are invited to " & **Invitation_Title** & return after field "Output"
put "The time is " & **Invitation_Time** & return after field "Output"
put "The date is " & **Invitation_Date** & return after field "Output"
```

In order for the program to know which data is a variable and which is text to be printed in a put statement, the ampersand `&` is used. This separates both the variable and the text to be printed on the screen as shown below:

```
put "You are invited to " & **Invitation_Title** & return after field "Output"
```
So, what have we learned so far?

The RevMedia program area has three areas:

1. The variable list - lists all variables used in the program
2. The event list - this is a list of all program blocks which are run when the button is clicked on by the user.
3. The components - contain the lines of code to be executed.

**ASK** is a command that allows the programmer to ask the user a question or ask the user for a response. For example:

```
ASK "Please enter the title of your party, for example, Easter Chicks Party"
```

**PUT** is a command that allows the programmer to transfer the users response (it) into a meaningful **variable**. For example:

```
PUT it into Invitation_Title
```

// are used to put internal commentary into a program or to space out different parts of the program to make it easier to read. For example:

```
// This block will print out the details of the party ticket
```

**On** and **End** are used to start and stop a block of program code. A program block must be started and ended, for example:

```
on Display_Invitation_Data
    put "You are invited to " & Invitation_Title & return after field "Output"
    put "The time of this party is " & Invitation_Time & return after field "Output"
    put "The date of this party is " & Invitation_Date & return after field "Output"
    put "Don't be late!!" & return after field "Output"
end Display_Invitation_Data
```

Well Done! You are now a computer programmer!
**Task 3: Address Labels Program**

**Specification**

A program is required in order to produce **6 address labels**. The details to be entered and displayed in a **loop** is the contact name, address, post code, city and phone number.

- **This is address label: 1**
  - Contact Name: Steven Whyte
  - Contact Address: 12 High Street
  - Contact Post Code: EH4 7FL
  - Contact City: Edinburgh
  - Contact Telephone Number: 01316656545

**Analysis**

I will do this task by first displaying a message on the screen asking for the user to enter the name. Once the user has entered this, I will then display a series of other messages asking the user for their address, post code, city and phone number. I will expect the user to key in these details also.

All details entered will then be displayed 6 times using a loop in a field called “output”.

**Design**

Here is the **design** showing the steps for the **Address Labels Program**:

**Stepwise Design** *(the main steps of the program)*:
1. Set up the variables
2. Get Contact Details
3. Display Contact Details

**Stepwise Refinement** *(breaking down the main steps into smaller steps)*:
1. **Set up the areas (variables) to store the user answers**
   1.1 set up the variables called contact_name, contact_address, contact_post_code, contact_city, contact_phone
2. **Get Contact Details**
   2.1 ask for the name of the user
   2.2 user puts in the name and this is stored in contact_name
   2.3 ask for the address of the user
   2.4 user puts in the address and this is stored in contact_address
   2.5 ask for the post code of the user
   2.6 user puts in the time and this is stored in invitation_time
   2.7 ask for the city of the user
   2.8 user puts in the date and this is stored in invitation_date
   2.9 ask for the telephone number of the user
   2.10 user puts in the telephone number and this is stored in contact_phone_number
3. **Display Contact Details**
   3.1 START a repeat with loop 1 to 6
   3.2 display the text Label Number followed by loop in the field called output
   3.3 display the text Contact Name followed by contact_name in the field called output
   3.4 display the text Contact Address followed by contact_address in the field called output
   3.5 display the text Contact Post Code followed by contact_post_code in the field called output
   3.6 display the text Contact City followed by contact_city in the field called output
   3.7 display the text Contact Phone followed by contact_phone in the field called output
   3.8 display a blank line
   3.9 END repeat
Task 3: Address Labels Program

Implementation

Use the analysis and the design on the previous page to help you complete this task. You will also find the code from Task 2 helpful as both of these programs are very similar.

Open the “Address Labels” template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 3...Address Labels.livecode

Like the previous task, double click on the “Create Address Label” button and enter the script for this button.

Test that your program works correctly by producing 6 address labels in the output box.

Save your program in your programming folder on the server as Task 3: Address Labels implementation testing.
Task 4: Apple Store

Specification

A program is required in order to work out the amount of money an Apple Store employee gets over the course of a week depending on their hourly wage and amount of hours worked.

The program should ask for the amount of hours worked over the week and then the hourly rate. It should then calculate the amount that they get a month and then a year and display this information on the screen along with a suitable message.

Analysis

I have been asked to create a program to work out the amount of money an Apple Store employee gets over the course of a week depending on their hourly wage and amount of hours worked.

I will do this task by first displaying a message on the screen asking for the user for their name. Once the user has entered this, I will then display another message asking for the number of hours they have worked. I will then ask the user for their hourly rate. Each answer will be placed into three variables.

In order to calculate the total pay, I will multiply the hours variable by the rate and place this into the variable total_pay.

The final step will involve displaying the users name and the total amount they have earned that week in a field called “output”

Design

Here is the design showing the steps for the Party Invitation Program:

Stepwise Design (the main steps of the program):
1. Set up the Variables (areas) to store the invitation details
2. Get Info
3. Calculate Total Pay
4. Display Total Pay

Stepwise Refinement (breaking down the main steps into smaller steps):
1. Set up the Variables (areas) to store the invitation details
   1.1 name_of_person, hours, rate, total_pay
2. Get Info
   2.1 ask for the users name
   2.2 user puts in their name and this is stored in name_of_person
   2.3 ask for the number of hours worked
   2.4 user puts in the hours worked and this is stored in hours
   2.5 ask for their hourly rate
   2.6 user puts in the hourly rate and this is stored in rate
3. Calculate Total Pay
   3.1 put hours multiplied by rate into the variable total_pay
4. Display Total Pay
   4.1 Put a message into the output field telling the name_of_person how much money they get using the variable total_pay
Task 4: Apple Store

Implementation

Open the “Apple Store” template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 4...Apple Store.livecode

Implementation (continued)

Assign the following code to the “Calculate Total Pay” button:

```livecode
on mouseUp
    // This event lists the program blocks that will occur when the button for 'Calculate Total Pay' has been pressed.
    get_info
    calculate_total_pay
    display_total_pay
end mouseUp

// This is a list of variables used in the Apple Store Program
local name_of_person, hours, rate, total_pay

on get_info
    // This component will get the users name, amount of hours worked that week and the users hourly rate.
    ask "Please enter your name, e.g. Steve Jobs: " put it into name_of_person
    ask "Please enter the number of hours you have worked in the Apple Store this week: " put it into hours
    ask "Please enter your hourly rate, e.g. 10.50 (no pound signs please!): " put it into rate
end get_info

on calculate_total_pay
    // This component will multiply the hours by the rate.
    // The answer the calculations will be placed in to total_pay.
    put hours * rate into total_pay
end calculate_total_pay

on display_total_pay
    // This component will display the users name and total pay in the field "Output".
    put return & "Well " & name_of_person & ": " & return after field "Output"
    // Set total_pay to be displayed as pounds and pence
    set numberformat to "00.00"
    put return & "This week, you have earned: £" & total_pay & return after field "Output"
end display_total_pay
```
Task 4: Apple Store

Testing (continued)

Testing is a very important part of any programming project. A program would not sell and would be useless if it did not produce the correct output when it was run.

Your Apple Store program is simple but must be tested to ensure that it produces the correct results.

It is advisable that you check your calculations using a calculator first. You should then run your program to check that you get the same results. This is a good indication that your program is indeed working correctly.

You should now run your program three times with data below. Enter the sample data for name, hours worked and hourly rate below and check that the results for “Total Pay” are the same as the results produced in your program.

<table>
<thead>
<tr>
<th>Apple Employee</th>
<th>Name</th>
<th>Hours Worked</th>
<th>Hourly Rate</th>
<th>Calculated Total Pay</th>
<th>Programs Total Pay</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Susan Wright</td>
<td>12</td>
<td>£8.00</td>
<td>£96.00</td>
<td>£96.00</td>
</tr>
<tr>
<td>2</td>
<td>Kevin Robertson</td>
<td>30</td>
<td>£10.00</td>
<td>£300.00</td>
<td>£300.00</td>
</tr>
<tr>
<td>3</td>
<td>Allan Stewart</td>
<td>39</td>
<td>£20.00</td>
<td>£780.00</td>
<td>£780.00</td>
</tr>
</tbody>
</table>

Save your program in your programming folder on the server as Task 4: Apple Store
Task 5: Edinburgh Running Club

Specification

The Edinburgh Running Club would like a simple program to calculate their team’s average ahead of a marathon. Your task is to design and produce a program that will take in the four team members’ names of Fred Smyth, Lauren Comb, Chris McKay and Steven Green along with the number of miles that had managed to run that week.

The team of four have agreed that if their group average is greater or equal to 26 miles, then they are ready to run the Great London Marathon. If not, then they are not and they must train together harder. You must find the average of all runs and then compare it with the marathon distance to make a decision.

Analysis

The first step is to set up the variables used in the program.

The second step is to take in the number of miles each that each runner has run over the course of the week.

The third step is to calculate the average number of miles that the group has run over the course of the week by adding each number of miles up and dividing by four.

The last step involves displaying each runners number of miles and then using the IF statement to decide whether or not the group are ready to run in the London Marathon. This will involve comparing the average number of miles with the marathon miles of 26.

Design

Here is the design showing the steps for the Edinburgh Running Club:

Stepwise Design (the main steps of the program):
1. Set up the variables
2. Set the Marathon Miles
3. Get the Marathon Miles
4. Calculate the Average Miles
5. Check and Display the Decision

Stepwise Refinement (breaking down the main steps into smaller steps):
1. Set up the areas (variables) to store the user answers
   1.1 set up the variables called Marathon Miles, Fred Miles, Lauren Miles, Chris Miles, Steven Miles, Average Miles
2. Set up the Marathon Miles
   2.1 Put the value 26 into the variable Marathon_Miles
3. Get Marathon Miles
   3.1 ask for the number of miles completed by Fred Smyth this week
   3.2 user puts in the amount of miles and this is stored in Fred_Miles
   3.3 ask for the number of miles completed by Lauren Comb this week
   3.4 user puts in the amount of miles and this is stored in Lauren_Miles
   3.5 ask for the number of miles completed by Chris McKay this week
   3.6 user puts in the time and this is stored in Chris_Miles
   3.7 ask for the number of miles completed by Steven Green this week
   3.8 user puts in the number of miles completed and this is stored in Steven_Miles
4. Calculate the Average Miles
   4.1 put Fred_Miles plus Lauren_Miles plus Chris_Miles plus Steven_Miles divided by 4 into the variable Average_Miles
5. Check and Display the Decision
   5.1 Set the number format to a whole number
   5.2 IF the average_Miles is greater or equal to the Marathon Miles THEN
   5.3 Display a message explaining that the group is ready for marathon with average_miles
   5.4 ELSE
   5.5 Display a message explaining that they are not ready for marathon with average_miles.
   5.6 END IF
Task 5: Edinburgh Running Club

Implementation

Open the “Edinburgh Running Club” template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 5...Edinburgh Running Club.livecode

Implementation

Assign the following code to the “Enter Miles” button:

```livecode
on mouseUp
  set_the_marathon_miles  
  get_the_marathon_miles  
  calculate_the_average_miles  
  check_and_display_the_decision  
end mouseUp

// Set up the global variables to be used in the Edinburgh Running Club Program
global Marathon_Miles, Fred_Miles, Lauren_Miles, Chris_Miles, Steven_Miles, Average_Miles

on set_the_marathon_miles
  put 26 into Marathon_Miles
end set_the_marathon_miles

on get_the_marathon_miles
  ask "Please enter the distance (in miles) that Fred Smyth has ran this week: "
  put it into Fred_Miles
  //
  ask "Please enter the distance (in miles) that Lauren Comb has ran this week: "
  put it into Lauren_Miles
  //
  ask "Please enter the distance (in miles) that Chris McKay has ran this week: "
  put it into Chris_Miles
  //
  ask "Please enter the distance (in miles) that Steven Green has ran this week: "
  put it into Steven_Miles
end get_the_marathon_miles

on calculate_the_average_miles
  // Find the average of the four distances by adding all distances up and then
  // dividing the answer by 4.
  // Put the answer in the variable called Average_Miles
  put (Fred_Miles + Lauren_Miles + Chris_Miles + Steven_Miles) / 4 into Average_Miles
end calculate_the_average_miles
```

The code is continued on the next page.
Task 5: Edinburgh Running Club

Implementation (continued)

on check_and_display_the_decision

// Set the any numbers displayed to a whole (integer) number
set numberformat to "0"

// Display all distances in the output_decision field
Put "Fred ran " & Fred_Miles & " miles this week." into line 1 of field Output_Decision
Put "Lauren ran " & Lauren_Miles & " miles this week." into line 2 of field Output_Decision
Put "Chris ran " & Chris_Miles & " miles this week." into line 3 of field Output_Decision
Put "Steven ran " & Steven_Miles & " miles this week." into line 4 of field Output_Decision
Put "The group's average miles ran this week is: " & Average_Miles & " miles." into line 6 of field Output_Decision

// This IF statement will decide whether or not the group are ready to run in the
// London Marathon by comparing their average_Miles with the Marathon_Miles. The
// Average_Miles will also be displayed as a whole number.
if Average_Miles >= Marathon_Miles then
  put "The group is ready to attempt the London Marathon! Good Luck!" into line 8 of field Output_Decision
else
  put "I am afraid that the group is not ready to attempt the London Marathon." into line 8 of field Output_Decision
end if
end check_and_display_the_decision

Testing

You should now test your program with the following sets of test data in the table below.

Remember - when it comes to your coursework, you will be expected to work out the answers using a
calculator first and then run your program to ensure you get the same answer.

<table>
<thead>
<tr>
<th>Names</th>
<th>Miles This Week</th>
<th>Calculated Average</th>
<th>Expected Decision</th>
<th>Program Average</th>
<th>Program Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred Smyth</td>
<td>26</td>
<td></td>
<td>Ready for Marathon</td>
<td>20</td>
<td>Ready for Marathon</td>
</tr>
<tr>
<td>Lauren Comb</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chris McKay</td>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Steven Green</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred Smyth</td>
<td>4</td>
<td>50</td>
<td>Not Ready for Marathon</td>
<td>20</td>
<td>Not Ready for Marathon</td>
</tr>
<tr>
<td>Lauren Comb</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chris McKay</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Steven Green</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred Smyth</td>
<td>26</td>
<td>26</td>
<td>Ready for Marathon</td>
<td>26</td>
<td>Ready for Marathon</td>
</tr>
<tr>
<td>Lauren Comb</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chris McKay</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Steven Green</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred Smyth</td>
<td>6</td>
<td>50</td>
<td>Not Ready for Marathon</td>
<td>20</td>
<td>Not Ready for Marathon</td>
</tr>
<tr>
<td>Lauren Comb</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chris McKay</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Steven Green</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Save your program in your programming folder on the server as
Task 5: Edinburgh Running Club
**Task 6: Cabins R Us**

**Specification**

A program has to be created to allow the user to choose a log cabin. Cabins can be booked for up to 7 days and come in three types and prices per night - Standard (£20), Premier (£50) and Luxury (£100).

The program will also ask for the number of occupants. It will then display the cost of the cabin over a 7 day period. If the customer rents a Luxury Cabin then they get a free complimentary bottle of champagne and food hamper with the rental.

**Analysis**

I have been asked to create a program which will allow a user to choose a log cabin.

The first step will involve setting up the variables to be used in the program.

The second step will involve getting the details of the cabin. The details will include the type of cabin the user is after followed by the number of occupants and the cost of the cabin per day.

The last step will involve putting a message on the screen displaying the cabin type, number of occupants and cabin cost. This will be placed into the field called output_details.

I will then use an IF statement to display a message informing the user that they get a complimentary bottle of champagne and a food hamper if the type of cabin ordered is a Luxury Cabin. This will be placed into output_details.

I will then start a loop for 1 to 7 days. Within this loop I will multiply the number of days by the cabin cost to show the cost of the cabin over a 7 day period. All of this information will be placed into the field called output_cost.

**Design**

The design (pseudocode) of the program is shown below. Study it carefully and make sure you know what is going on:

**Stepwise Design**

1. **Set up the variables**
2. **Get Cabin Details**
3. **Display Cabin Details**

**Stepwise Refinement**

1. **Set up the variables**
   1.1 Set `cabin_type`, `cabin_occupants`, `cabin_cost` as global variables

2. **Get Cabin Details**
   1.1 Ask the user for the cabin type (Standard, Premier or Luxury)
   1.2 Put the result into the variable `cabin_type`
   1.3 Ask the user for the number of people staying in the cabin
   1.4 Put the result into the variable `cabin_occupants`
   1.5 Ask the user for the cabin cost per day (20, 50 or 100)
   1.6 Put the result into the variable `cabin_cost`

3. **Display Cabin Details**
   2.1 Put a message onto the screen with the type of cabin ordered showing variable cabin type into a field called Output Details
   2.2 Put a message onto the screen with the number of people staying showing variable cabin occupants into a field called Output Details
   2.3 Put a message onto the screen with the cost of the of cabin showing variable cabin cost into a field called Output Details
   2.4 If the cabin type is equal to Luxury then put a message onto the screen telling the user that they get a free complimentary bottle of champagne and a food hamper.
   2.5 Start a Repeat loop 1 to 7 days
   2.6 Put a message onto the screen with the day number and day * cabin cost into a field called Output Cost
   2.7 End the Repeat loop
Task 6: Cabins R Us

Implementation - Creating the user interface

Open the “Cabins R Us” template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 6...Cabins R Us.livecode

Create two buttons named “Get Details” and “Clear” as shown in the template stack below.

Create two output fields named “Output_Details” and “Output_Cost” as shown above.

Once you have created the buttons and the output boxes, enter the code on the next page into the “Get Details” button.

You will also need to enter a simple piece of code for the “Clear” button to clear the contents of the “Output_Details” and “Output_Cost” fields. Look back at Task 5 (page 17) to help you write the script for the “Clear” button.
Task 6: Cabins R Us

Implementation - Creating the program code

You are now ready to begin developing the program script for the “Get Details” button. This program is below and should be keyed carefully.

The teacher will go over the program with you below before you being the coding.

Write a script for the “Clear” button also. Use the previous program to help you write the simple script to clear the “Output_Details” and the “Output_Cost” fields.

Test that your program is working correctly using the test data supplied on the next page.

```livecode
on mouseUp
  get_details
  display_details
end mouseUp

global cabin_type, cabin_occupants, cabin_cost, day

on get_details
  ask "Please enter cabin type (Standard, Premier, Luxury): "
  put it into cabin_type
  ask "Please enter number of occupants: "
  put it into cabin_occupants
  ask "Please enter cost per day (20, 50, 100): "
  put it into cabin_cost
end get_details

on display_details
  put "You have ordered a " & cabin_type & " cabin." & return after field "Output_Details"
  put "You have told us that there are " & cabin_occupants & " people staying in this cabin." & return after field "Output_Details"
  put "The cabin cost is £" & cabin_cost & " per night." & return after field "Output_Details"
  if cabin_type = "Luxury" then
    put "You get a free complimentary bottle of champagne and food hamper with the Luxury Cabin." & return after field "Output_Details"
  end if
  repeat with day = 1 to 7
    put "Day " & day & " £" & day * cabin_cost & return after field "Output_Cost"
  end repeat
end display_details
```

Create the Clear button

Once you have keyed in all of the code for the “Get Details” button, create the script to clear the “Output_Details” and “Output_Cost” fields when the Clear button is pressed. Use the program on page 17 to help you if you have forgotten.

Test that your program produces the correct output using the test data supplied on the next page.
Task 6: Cabins R Us

Testing

You should now fully test your program with the test data below.

<table>
<thead>
<tr>
<th>Cabin Type</th>
<th>Number of Occupants</th>
<th>Cabin Cost</th>
<th>Expected Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>4</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 Day</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 Days</td>
</tr>
<tr>
<td>Premier</td>
<td>8</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 Day</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 Days</td>
</tr>
<tr>
<td>Luxury</td>
<td>2</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 Day</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6 Days</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>7 Days</td>
</tr>
</tbody>
</table>

Evaluation

Fit for Purpose

The Cabins R Us Program is fit for purpose. It successfully asks for and takes in the type of cabin, number of occupants and cabin cost. The program also successfully works out the cost of the cabin over a seven day period and displays the results on the screen.

The program makes use of an IF statement to decide whether or not the customer gets a complimentary bottle of champagne and food hamper if they select a Luxury Cabin. This also works correctly as shown above.

Suggested Improvements

The program could be improved by asking the user which dates they would like to hire the cabin from. The program could also include a picture of the cabin the customer intends to hire along with the exact location of the cabin.

Save your program in your programming folder on the server as Task 6: Cabins R Us
Task 7: Who's on Detention?

Specification

A program has to be created to work out the number of pupils who should be placed on a late's detention.

A pupil is placed on a late’s detention IF their number of late’s over Monday to Friday is greater than the lates_trigger (2). This will trigger that the pupil has a detention to complete.

The program will then display the number of pupils who have been placed on a late's detention.

Analysis

I have been asked to create a program which will work out the number of pupils who should be placed on late's detention.

The first step will involve setting up the variables to be used in the program.

The second step will involve setting up a loop for the 8 pupils. For each pupil, the user will be prompted for the number of times (if any) they were late to registration over the course of the five day week.

If the value is greater than the number of late’s required for detention (2) then one will be added to the detentions count.

The third and final step will involve displaying the number of pupils of the 8 that should be placed on detention.

Design

The design (pseudocode) of the program is shown below. Study it carefully and make sure you know what is going on:

Stepwise Design
1. Set up the variables
2. Enter Data
3. Display Result

Stepwise Refinement
1. Set up the variables
   1.1 Set lates, loop, lates_trigger, and detentions as global variables
   1.2 Put the value 0 into the variable detentions
   1.3 Put the value 2 into the variable lates_trigger

2. Enter Data
   1.1 Start a Repeat loop 1 to 8 pupils
   1.2 Prompt the user for the number of lates for the pupil
   1.3 Put the value lates into a new line in the field called output_number
   1.4 IF the value of lates is greater than the lates_trigger then add 1 to detentions variable
   1.5 End Repeat

3. Display Result
   3.1 Display a message to say that the number of pupils on detention this week is: detentions into the field called output_result
Task 7: Who’s on Detention?

Implementation - Creating the user interface

Open the “Who's on Detention” template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 7...Who’s on Detention.rev

‘Create two buttons and label them “Number of Lates” and “Clear”.

Once your program buttons have been added, your stack should look similar to the screenshot shown below.

Implementation - Creating the program code

You are now ready to begin developing the program script for the “Number of Lates” button. This program is on the next page and should be keyed carefully into the script for “Number of Lates”.

You should also create a script for the “Clear” button. Use a previous program (page 17) to help you create the simple script to clear both the “Output_Number” and “Output_Result” fields.
Task 7: Who’s on Detention

Implementation - Creating the program code

on mouseUp
  set_up
  enter_data
  display_result
end mouseUp

global lates, loop, lates_trigger, detentions

on set_up
  put 0 into detentions
  put 2 into lates_trigger
end set_up

on enter_data
  // Set up a fixed loop of 8 pupils
  repeat with loop = 1 to 8
    ask "Please enter the number of lates by pupil number " & loop
  
  // If the cancel button is pressed, go back to the start of the program.
  if the result = "Cancel" then
    exit to top
  end if

  put it into lates
  put lates into line loop of field "output_number"
  // If the number of lates entered by the user is greater than the lates trigger
  // then add 1 to the variable detentions.
  IF lates > lates_trigger THEN add 1 to detentions
end repeat
end enter_data

on display_result
  put "The number of pupils on detention this week is " & detentions into field "output_result"
end display_result

Create the Clear button

Once you have keyed in all of the code for the “Get Details” button, create the script to clear the “Output_Number” and “Output_Result” fields when the Clear button is pressed. Use the program on page 17 to help you if you have forgotten.

Test that your program produces the correct output using the test data supplied on the next page.
**Task 7: Who’s on Detention**

**Testing**

Test that your program produces the correct output using the test data supplied below.

<table>
<thead>
<tr>
<th>Number of Late’s by Each Pupil</th>
<th>Expected Number of Detentions</th>
<th>Programs Number of Detentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, 1, 0, 0, 2, 0, 1, 1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2, 3, 2, 2, 3, 3, 5, 4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3, 3, 5, 4, 4, 5, 4, 3</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

**Evaluation**

**Fit for Purpose**

The Who’s on Detention Program is fit for purpose. It successfully asks for and takes in the number of late’s for each of the 8 pupils. It will then successfully compare the lates value with the late’s trigger and if it is over two, a detention will be issued. The program issues the correct number of detentions and this has been shown in the testing table above.

**Suggested Improvements**

The program could be improved upon by asking the user for the names of the pupils on detention so teachers knew who was going to be turning up.

---

Save your program in your programming folder on the server as Task 7: Whos on Detention

Congratulations!
You are now ready to move onto the two General Coursework exercises.
Task 8: High, Low, Spot On!

Specification

The computer has chosen a secret number which is between 0 and 100. You have to try to guess what it is.

The program you will create will take in the users guess and decide if the guess is lower, higher or equal to that of the random number the computer is thing of.

A message of “Too High”, “Too Low” or “Spot On” will be displayed in the programs output field. If the users guess is too high or too low, they will be given the opportunity to re-enter another number.

This simple game will then count up the number of guesses and display this number in the programs output field once the user has correctly guessed the computers number.

Analysis

I have been asked to create a program which will allow a user to try and guess a randomly generated number between 0 and 100.

I will do this task in four steps.

The first step will involve setting up the variables to be used in the program.

The second step is to put a randomly generated number into the variable my_number.

The third step will involve asking the user to enter a whole number between 0 and 100. This will be validated using the validation function.

The fourth step will involve comparing each guess the user makes with the randomly generated number that has been chosen and displaying a suitable message of “Too High”, “Too Low” or “Spot On”.

Every valid guess will be counted and an output message will be displayed to the user information them of how many guesses it took them to find the correct randomly generated number.
Task 8: High, Low, Spot On!

<table>
<thead>
<tr>
<th>CREDIT ONLY</th>
<th>CREDIT ONLY</th>
<th>CREDIT ONLY</th>
<th>CREDIT ONLY</th>
<th>CREDIT ONLY</th>
<th>CREDIT ONLY</th>
</tr>
</thead>
</table>

### Design

Here is the **design** showing the steps for the High, Low, Spot On Program:

1. **Set up the areas (variables) to store the data for the game**
   1.1 set up variable called guess_count
   1.2 set up variable called my_number
   1.3 set up variable called guess

2. **Set up game**
   2.1 Put the value 0 into the variable guess_count
   2.2 Put a random number between 0 and 100 into the variable my_number

3. **Play game**
   3.1 **Start a repeat loop** until the users guess is equal to my_number
   3.2 ask for the users guess between 0 and 100.
   3.3 If the cancel button is pressed
   3.4 Exit to the top of the program
   3.5 **End if**
   3.6 put the result into the variable guess
   3.7 put guess and a carriage return before the field output_number
   3.8 If users guess is greater than my_number then put “Too High” and a carriage return before the field output_decision
   3.9 If users guess is less than my_number then put “Too Low” and a carriage return before the field output_decision
   3.10 If users guess is equal to my_number then put “Spot On!” and a carriage return before the field output_decision
   3.11 add 1 to the variable guess_count after each guess
   3.12 **End repeat loop**
   3.13 Put users guess_count into field output_guess

### Implementation

Open the “High Low Spot On” template. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 8...High Low Spot On.livecode
Before you begin creating the program script, you need to create two buttons to assign the program to and two output boxes to display the program output. The first button will be called “Play Game” and the second button will be called “Clear”. This will clear the output fields (white boxes) so the game can be played again.

Make sure you are in “Edit Mode”. Drag the button object on to the stack and double click it and change the Name, Label and Tool tip to “Play Game”. Do the same with the “Clear” button.

You should now add two scrolling output fields which will display different information in the game. To do this, drag the output boxes onto the stack and position them in a similar way as shown above. You now need to name the output fields so that you can use them in your program script. Double click on the top output field and call it “output_decision” and then bottom field and call it “output_guess”.

Once you have added and named your buttons and output fields, your stack should look like the one above. You are now ready to begin creating your script for the High, Low Spot on program.

You will also be shown the script to clear the output fields (white boxes). Remember this script as you will use it in future programs.
Task 8: High, Low, Spot On!

**Implementation**

Assign the following code to the “Play Game” button:

```livecode
// Set up the variables
global guess_count, my_number, guess

on mouseUp
    set_up
    play_game
end mouseUp

on set_up
    // The variable "guess_count" will count the number of guesses the user has taken.
    put 0 into guess_count
    // Set up a variable called my_number. This will be the number that the user
    // will try to guess.
    // Make the my_number variable a random number up to 100.
    put random (100) into my_number
end set_up

on play_game
    // This loop will keep repeating until the user guesses the correct number.
    repeat until guess = my_number
        ask "Please enter your guess between 0 and 100"
        // If the cancel button is pressed, go back to the start of the program.
        if the result = "Cancel" then
            exit to top
        end if
        // Put the users guess into the variable guess
        put it into guess
        // The IF statements will compare the users guess with the programs
        // randomly generated number and display an appropriate message.
        if guess>my_number then put guess & " --> Too High" & CR BEFORE fld "output_decision"
        if guess<my_number then put guess & " --> Too Low" & CR BEFORE fld "output_decision"
        if guess=my_number then put guess & " --> Spot On!!" & CR BEFORE fld "output_decision"
        // Once the user has had a guess, 1 will be added to the variable "guess_count".
        add 1 to guess_count
    end repeat
    put "You found the number in " & guess_count & " guesses." into field "output_guess"
end play_game
```

Assign the following code to the “Clear” button:

```livecode
on mouseUp
    // This program will put "empty" into the output fields
    put empty into field "output_decision"
    put empty into field "output_guess"
end mouseUp
```
Task 8: High, Low, Spot On!

You should now test your program with similar test data to that below. The test data table below assumes that the number 8 has to be found.

You should test your program with similar test data.

<table>
<thead>
<tr>
<th>Guess</th>
<th>Output</th>
<th>Number of Guesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>Too High</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>Too High</td>
<td>2</td>
</tr>
<tr>
<td>60</td>
<td>Too High</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>Too Low</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>Too Low</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>Too High</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>Too High</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>Too High</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>Spot on!</td>
<td>9</td>
</tr>
</tbody>
</table>

Save your program in your programming folder on the server as Task 8: High Low Spot On.
Task 9: Computing Exam Marks Program

Specification

A program is to be written to work out whether or not a student has passed the theory part of the Standard Grade Computing Course. The program should compare the students overall percentage with the passmark percentage of 50%.

The program will be required to take in a pupils name along with their five computing results for each of the topics that are covered at Standard Grade level.

Each test is out of 100 and each result entered must be validated as being a whole number between 0 and 100.

Once each mark has been entered, the student’s overall percentage mark should be worked out.

Finally, the percentage mark should be compared with the passmark percentage of 50%. If the percentage mark is greater or equal to the passmark percentage, the student should pass the course, else the student should fail the course.

Analysis

I have been asked to create a program which will work out whether or not a computing student has passed or failed the Standard Grade Computing Course.

I will first set up the variables to be used in the program and then set up the passmark percentage to hold the value of 50 as a pass in the course is 50% or more.

I will then ask the user for the student’s name followed by each of their marks in the five computing exams. Each mark will be validated as a whole number between 0 and 50.

Once each mark has been entered, I will work out the percentage by adding up each of the five marks, dividing by 500 (which is the total of all marks added together), and then multiplying the answer by 100. This answer will be placed into the percentage mark variable.

The final part of this program will involve using an IF statement to make a decision as to whether or not the student has passed or failed the course. I will do this by comparing the passmark percentage with the percentage mark.

A suitable message will be displayed of passed or failed depending on the students percentage mark along with their total percentage.
Task 9: Computing Exam Marks Program

### Design

Remember, the design of a program is very important as it allows the programmer to think about the structure of the program before the code is produced.

As mentioned earlier, the design of a program is known as Pseudocode. Pseudocode is a cross between programming language and our own English language and makes a program easier to understand without relying on the use of a programs complex commands.

The design is built up of two parts, the first is the **Stepwise design**. This shows the main steps of the program. The second part is the **Stepwise Refinement**. This involves breaking these main steps into even smaller steps so it can be taken and programmed in any programming environment.

The following design is for **Computing Exam Marks Program**. Study it closely as you will need to produce a similar design for your programs when it comes to the practical coursework.

#### Stepwise Design

1. set up the variables
2. set the passmark
3. get_student_name
4. get and validate the marks
5. calculate percentage mark
6. check and display the decision
7. validate

#### Stepwise Refinement

1. Set up the variables
   1.1 Set Passmark_Percentage, Student_Name, Check_Number, CS_Mark, GPP_Mark, CN_Mark, CDP_Mark, IA_Mark, Percentage_Mark as global variables
2. set the passmark
   2.1 set Percentage_Passmark to 50
3. get student name
   3.1 ask for the name of the student
   3.2 user puts in the student’s name and this is stored in Student_Name
4. get and validate the marks
   4.1 ask for the Student_Name’s mark in Computer Systems (0-100)
   4.2 put the mark entered into Check_Number
   4.3 Start the validation function
   4.4 Put the result of Check_Number into CS_Mark
   4.5 ask for the Student_Name’s mark in General Purpose Packages (0-100)
   4.6 put the mark entered into Check_Number
   4.7 Start the validation function
   4.8 Put the result of Check_Number into GPP_Mark
   4.9 ask for the Student_Name’s mark in Communications and Networks (0-100)
   4.10 put the mark entered into Check_Number
   4.11 Start the validation function
   4.12 Put the result of Check_Number into CN_Mark

The design for the program is continued on the next page
Task 9: Computing Exam Marks Program

### Design (continued)

4.13 ask for the Student_Name’s mark in Commercial Data Processing (0-100)
4.14 put the mark entered into Check_Number
4.15 Start the validation function
4.16 Put the result of Check_Number into CDP_Mark

4.13 ask for the Student_Name’s mark in Commercial Data Processing (0-100)
4.14 put the mark entered into Check_Number
4.15 Start the validation function
4.16 Put the result of Check_Number into CDP_Mark

4.17 ask for the Student_Name’s mark in Industrial Applications (0-100)
4.18 put the mark entered into Check_Number
4.19 Start the validation function
4.20 Put the result of Check_Number into IA_Mark

5. calculate the percentage mark
5.1 put the CS_Mark plus the GPP_Mark plus the CN_Mark plus the CDP_Mark plus the IA_Mark divided by 500 multiplied by 100 into the variable Percentage_Mark

6. check and display the marks
6.1 Set the number format of any number displayed to a whole number
6.2 Display the CS_Mark entered into line 1 of the Output_Result field
6.3 Display the GPP_Mark entered into line 2 of the Output_Result field
6.4 Display the CN_Mark entered into line 3 of the Output_Result field
6.5 Display the CDP_Mark entered into line 4 of the Output_Result field
6.6 Display the IA_Mark entered into line 5 of the Output_Result field
6.7 If the Percentage_Mark is greater or equal to the Percentage_Passmark then
6.8 Display message Student_Name has passed with a Percentage_Mark in line 1 of the field called Output_Decision
6.9 Else
6.10 Display message Student_Name has failed with: Percentage_Mark in line 1 of the field called Output_Decision
6.11 End If

7. Validate
7.1 Start a Repeat Loop until Check_Number is greater or equal to 0 and less than or equal to 100 and is a whole number (integer)
7.2 Ask the user to re-enter a whole number between 0 and 100 if the data entered is invalid
7.3 If the cancel button is pressed
7.4 Exit to the top of the program
7.5 End If
7.6 Put the result into Check_Number
7.7 End Repeat Loop

### Implementation - Creating the user interface

Open the “Computing Exam Marks” template.

It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 9...Computing Exam Marks.livecode
Task 9: Computing Exam Marks Program

Implementation - Creating the user interface (continued)

Create two buttons and label them “Enter Marks” and “Clear”. Create two scrolling fields and name it “Output_Results” and “Output_Decision”.

Once your programs objects have been added, your stack should look similar to the screenshot shown below.

Implementation - Creating the program code

You are now ready to begin developing the program script for the “Enter Marks” button. This program is on the next page and should be keyed carefully into the script for Enter Marks.

You should also create a script for the “Clear” button. Use the previous programs to help you create the simple script to clear the “Output_Results” and “Output_Decision” fields.
Task 9: Computing Exam Marks Program

Implementation - Creating the program code (continued)

Assign the following script to your “Enter Marks” button.

```livecode
on mouseUp
    set_the_passmark
    get_student_name
    get_and_validate_the_marks
    calculate_the_percentage_mark
    check_and_display_the_decision
    validate
end mouseUp

// Set up the global variables to be used in the Maths and English Marks program
global Passmark_Percentage, Student_Name, Check_Number, CS_Mark, GPP_Mark, CN_Mark, CDP_Mark, IA_Mark, Percentage_Mark

on set_the_passmark
    // Assign it the value of 50 to the variable Passmark_Percentage.
    put 50 into Passmark_Percentage
end set_the_passmark

on get_student_name
    ask "Please enter the student name: "
    put it into Student_Name
end get_student_name

on get_and_validate_the_marks
    // Get the five marks between 0 and 100.
    // Put each mark entered into the variable "Check_Number" so that it can be validated as being between 1 and 100.
    // Start the validation function at the bottom of the program
    // Once the value has been checked as valid, put the value into the variable.
    ask "Please enter " & student_name & "'s mark for Computer Systems (0–100)"
    put it into Check_Number
    validate
    put Check_Number into CS_Mark
    //
    ask "Please enter " & student_name & "'s mark for General Purpose Packages (0–100)"
    put it into Check_Number
    validate
    put Check_Number into GPP_Mark
    //
    ask "Please enter " & student_name & "'s mark for Communications and Networks (0–100)"
    put it into Check_Number
    validate
    put Check_Number into CN_Mark
    //
    ask "Please enter " & student_name & "'s mark for Commercial Data Processing (0–100)"
    put it into Check_Number
    validate
    put Check_Number into CDP_Mark
    //
    ask "Please enter " & student_name & "'s mark for Industrial Applications (0–100)"
    put it into Check_Number
    validate
    put Check_Number into IA_Mark
end get_and_validate_the_marks
```

The code is continued on the next page
Task 9: Computing Exam Marks Program

Implementation - Creating the program code (continued)

```livecode
on calculate_the_percentage_mark
  // Find the percentage of the five marks by adding all exam marks up and then
  // dividing the answer by the total of all the exams then multiplying the answer by 100.
  // Put the answer in the variable called Percentage_Mark
  put (CS_Mark + GPP_Mark + CN_Mark + CDP_Mark + IA_Mark) / 500 * 100 into Percentage_Mark
end calculate_the_percentage_mark

on check_and_display_the_decision
  // Display Maths and English marks in the output field.
  // This IF statement will decide whether or not the student has passed by taking the average
  // mark and comparing it with the passmark. The percentage mark will also be displayed as a
  // whole number.
  set numberformat to "0"
  put CS_Mark into line 1 of field Output_Results
  put GPP_Mark into line 2 of field Output_Results
  put CN_Mark into line 3 of field Output_Results
  put CDP_Mark into line 4 of field Output_Results
  put IA_Mark into line 5 of field Output_Results
  if Percentage_Mark >= Passmark_Percentage then
    put Student_Name & " has passed the Computing Course with: " & Percentage_Mark & "%" into line 1 of field Output_Decision
  else
    put Student_Name & " has failed the Computing Course with: " & Percentage_Mark & "%" into line 1 of field Output_Decision
  end if
end check_and_display_the_decision

on validate
  // Check that the number entered is between 0 and 100 and is a whole number (integer).
  // If the number entered is not in this range or is not a whole number, the user is asked to re-
  // enter. If the cancel button is pressed, go back to the start of the program.
  repeat until Check_Number >= 0 and Check_Number <= 100 and Check_number is an integer
    ask "You have entered an invalid mark. Please re-enter your mark between 0 to 100"
    if the result = "Cancel" then
      exit to top
    end if
    put it into Check_Number
  end repeat
end validate
```

Testing

You should now test your program with the test data on the next page.

Ensure that the validation function does not accept the first five sets of numbers in the table and then make sure that the rest of the valid data is accepted producing the correct average mark and decision.

An evaluation for this program has been completed. The evaluation involves looking back at the program and ensuring that it does everything that it was supposed to do (fit for purpose) as well as suggesting any improvements.
Task 9: Computing Exam Marks Program

Testing (continued)

<table>
<thead>
<tr>
<th>CS Mark</th>
<th>GPP Mark</th>
<th>CN Mark</th>
<th>CDP Mark</th>
<th>IA Mark</th>
<th>Calculated Percentage Mark</th>
<th>Program Percentage Mark</th>
<th>Course Decision (Pass/Fail)</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Invalid CS Mark</td>
<td>Invalid CS Mark</td>
<td>N/A</td>
</tr>
<tr>
<td>12</td>
<td>1001</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>Invalid GPP Mark</td>
<td>Invalid GPP Mark</td>
<td>N/A</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>999</td>
<td>N/A</td>
<td>N/A</td>
<td>Invalid CN Mark</td>
<td>Invalid CN Mark</td>
<td>N/A</td>
</tr>
<tr>
<td>99</td>
<td>100</td>
<td>76</td>
<td>10.5</td>
<td>N/A</td>
<td>Invalid CDP Mark</td>
<td>Invalid CDP Mark</td>
<td>N/A</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>-10</td>
<td>Invalid IA Mark</td>
<td>Invalid IA Mark</td>
<td>N/A</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>10</td>
<td>100</td>
<td>100</td>
<td>100%</td>
<td>100%</td>
<td>PASS</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
<td>60</td>
<td>50</td>
<td>50</td>
<td>44%</td>
<td>44%</td>
<td>FAIL</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>1</td>
<td>10</td>
<td>20</td>
<td>9%</td>
<td>9%</td>
<td>FAIL</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>45</td>
<td>60</td>
<td>65</td>
<td>54%</td>
<td>54%</td>
<td>PASS</td>
</tr>
</tbody>
</table>

Evaluation

Fit for Purpose

The Maths and English Marks Program is fit for purpose. It successfully validates the five marks as being whole numbers between 0 and 100.

The program also successfully works out the course percentage as can be seen with the table above. I worked out the percentages using a calculator and then ran my program with the test data to ensure that the same results were given.

The program makes use of an IF statement to decide whether the student has passed or failed the course based on the percentage as shown above.

Suggested Improvements

The program can be improved by perhaps taking in the students practical grades so it gave them their overall grade in practical coursework tasks as well as the theory exams.

Save your program in your programming folder on the server as Task 9: Computing Exam Marks Marks
Task 10: Music Shop Takings (Logical Operators)

Specification
A music store sells CD’s, DVD’s and Blu-Ray disks. The program will take in the total number of CD’s, DVD’s and Blu-Ray Disks sold over the course of the working day. The program will then find the combined total cost of CD’s (£7.99), DVD’s (£10.99) and Blu-Ray Disks (£14.99) sold.

- If the employee sells 30 or more CD’s, 35 or more DVD’s and 35 or more Blu-Ray Disks, they are entitled to a bonus.
- ELSE
  - They have not made enough money on today’s takings to be eligible for a bonus.

Design
The design of the program is shown below:

Stepwise Design
1. Get Number of Items Sold
2. Calculate Total Cost
3. Decide

Stepwise Refinement
1. Get Number of Items Sold
   1.1 Ask the user for the number of cd’s sold
   1.2 Put the result into the variable CDs_Sold
   1.3 Put CDs_Sold into line 1 of the output total field
   1.4 Ask the the user for the number of DVD’s sold
   1.5 Put the result into the variable DVDs_Sold
   1.6 Put DVDs_Sold into line 2 of the output total field
   1.7 Ask the the user for the number of Blu-Ray’s sold
   1.8 Put the result into the variable BluRays_Sold
   1.9 Put BluRays_Sold into line 3 of the output total field

2. Calculate Total Cost
   2.1 Put the CDs_Sold multiplied by £7.99 plus the DVDs_Sold multiplied by £10.99 plus the BluRays_Sold multiplied by £14.99 into a variable called Total_Cost

3. Decide
   3.1 set the format to 00.00
   3.2 Put a message onto the screen telling the user how much they have made today. Show Total_Cost variable. Put this message into line 1 of field “Output_Decision”
   3.3 If the CDs_Sold is greater or equal to 30 AND the DVDs_Sold is greater or equal to 40 OR the BluRays_Sold is greater or equal to 50 THEN
   3.4 Put a message on the screen saying, Wow, what a profit! Put this message into line 2 of field “Output_Decision”
   3.5 Else
   3.6 Put a message on the screen saying that they have not sold enough to make a profit on today’s takings. Put this message into line 2 of field “Output_Decision”
   3.7 End If
**Task 10: Music Shop Takings (Logical Operators)**

**Implementation**

Open the “Music Shop Takings” program. It can be found in:

Resources > Computing > Standard Grade > Practical Work > Programming > 10...Music Shop Takings.livecode

Assign the code below to the “Go” button.

**Implementation (continued)**

Here is the code for the program. Pay special attention to the decide function as it uses the logical operators of **AND** and **OR** in order to decide whether:

A large profit has been made and the user deserves a bonus

**OR**

No profit has been made and no bonus is given.

```livecode
global CDs_Sold, DVDs_Sold, BluRays_Sold, Total_Cost

on mouseUp
    get_number_of_items_sold
    calculate_total_cost
    decide
end mouseUp

on get_number_of_items_sold
    // Get the number of CDs, DVDs and Blu-Ray Disks sold
    ask "Please enter the number of CD's sold today: "
    put it into CDs_Sold
    put CDs_Sold into line 1 of field "output_total"
    ask "Please enter the number of DVD's sold today: "
    put it into DVDs_Sold
    put DVDs_Sold into line 2 of field "output_total"
    ask "Please enter the number of Blu-Ray's sold today: "
    put it into BluRays_Sold
    put BluRays_Sold into line 3 of field "output_total"
end get_number_of_items_sold

on calculate_total_cost
    // Calculate the total cost by taking the amount of items sold by the user and multiplying it by
    // the of each product
    put CDs_Sold * 7.99 + DVDs_Sold * 10.99 + BluRays_Sold * 14.99 into Total_Cost
end calculate_total_cost

on decide
    // This function sets the output of the total cost to pounds and pence
    set numberformat to "00.00"
    // Display the total cost
    put "Today you have made a total of £" & Total_Cost into line 1 of field "output_decision"
    // Logical operators are used with amounts to determine what output message is displayed and
    // whether or not the employee deserves a special bonus
    if CDs_Sold >= 30 and (DVDs_Sold >= 40 or BluRays_Sold >= 50) then
        put "Wow, what a profit! You deserve a special bonus!" into line 2 of field "output_decision"
    else
        put "You have not sold enough to make a profit today." into line 2 of field "output_decision"
    end if
end decide
```
Task 10: Music Shop Takings (Logical Operators)

Testing

<table>
<thead>
<tr>
<th>CD’s Sold (£7.99)</th>
<th>DVD’s Sold (£10.99)</th>
<th>Blu-Ray’s Sold (£14.99)</th>
<th>Calculated Output (use a calculator)</th>
<th>Program Output (run your program)</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>50</td>
<td>50</td>
<td>Total Cost: £1530.71 &quot;You have not sold enough to make a profit today.&quot;</td>
<td>Total Cost: £1530.71 &quot;You have not sold enough to make a profit today.&quot;</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>Total Cost: £169.85 &quot;You have not sold enough to make a profit today.&quot;</td>
<td>Total Cost: £169.85 &quot;You have not sold enough to make a profit today.&quot;</td>
</tr>
<tr>
<td>30</td>
<td>41</td>
<td>34</td>
<td>Total Cost: £1199.95 &quot;Wow, what a profit!&quot;</td>
<td>Total Cost: £1199.95 &quot;Wow, what a profit!&quot;</td>
</tr>
<tr>
<td>35</td>
<td>28</td>
<td>51</td>
<td>Total Cost: £1351.86 &quot;Wow, what a profit!&quot;</td>
<td>Total Cost: £1351.86 &quot;Wow, what a profit!&quot;</td>
</tr>
</tbody>
</table>

If both answers produced from the “Expected Output” and “Actual Output” match then you know that your program is working correctly.

Evaluation

Fit for Purpose

The Music Shop Takings Program is fit for purpose. It successfully asks for and takes in the number of CD’s, DVD’s and Blu-Ray Disks sold and displays the total cost on the screen for the user to see.

The program successfully makes use of an IF statement with logical operators to decide whether or not the employee gets a bonus if they sell over a certain number of CD’s, DVD’s and Blu-Ray Disks. This also works correctly as shown in the testing table above.

Suggested Improvements

The program can be improved by asking the user what their name is so that the manager knows which employee deserves the sales bonus.

Save your program in your programming folder on the server as Task 10: Music Shop Takings

Congratulations!
You are now ready to move onto the two Credit Coursework exercises.