Programming With

LIVECODE Community Edition

What can I make with LiveCode?

LiveCode is used to create apps, games, interactive ebooks and comics.

LiveCode is used to create powerful in-house systems and mobile apps for public and private sector organizations.

National 4 Computing Science

Materials produced at GHS
By Mr S. whyte

The Software Development Process

Introduction

The Software Development Process (**SDP**) can be split into **7 main** steps which are carried out in **order**. These steps should be carried out when creating **any** software project and are summarised below.

Analysis A

A statement about **what** your program is going to do. You will also include a description of the **main steps** of the problem.

Design Dance

This involves designing both the **user interface** and the **structure** of the **program code**.

For the purpose of Intermediate 2 Computing, more emphasis will be placed on designing the **structure** of the **program code** rather than the design of the user interface. We will be using a design notation known as **pseudocode** to achieve this. More is mentioned about pseudocode on the next page.

Implementation In

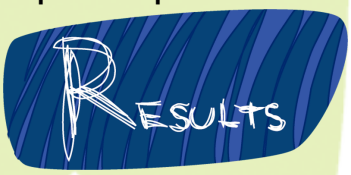
The implementation stage involves keying in the program code using the built in **text editor** within the programming environment. We will use **LiveCode** to create our programs.

Testing The

Testing is an important part of any project. Testing ensures that your program is **reliable** and **robust** in the sense that it should produce the **correct results** and **not crash** due to **unexpected input**.

We should test our program with **three** sets of test data. These are:

- **Normal** (accepted data within a set range)
- **Extreme** (accepted data on the boundaries)
- **Exceptional** (data that is not accepted).



Documentation Dark

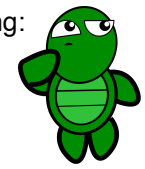
Documentation is usually produced in the form of a **user guide** and a **technical guide**. The user guide shows the user how to **use** the **functions** and **features** of the **software** whereas the **technical guide** gives the user information on how to **install** the **software** as well as the **minimum system requirements**.

Evaluation Every

An evaluation is usually a **review** which shows that your program is **fit for purpose**, in other words, it does **exactly** what it was **designed** to do.

The evaluation should also focus on the **readability** of your program code. For example, if **another programmer** was asked to **maintain** your program code at a later date, would they be able to understand what was going on? You should always ensure your program is **readable** by doing the following:

- **Use** of **meaningful identifiers** for **variable** and **array names**
- **Use** of **internal commentary** *(// This subroutine will do the following....)*
- **Effective** use of **white space** between **subroutines** to **space out** the **program**.
- **Indentation** to show the **start** and **end** of any control structures such as a fixed **loop**.



Maintenance Monday

Maintenance is performed at the **very end** of the project. You will not be required to perform any maintenance on your programs but you will need to know about **Corrective**, **Adaptive** and **Perfective** maintenance. These are covered in the Software Development theory notes.

The Design Process

Pseudocode

The design of a program is **very important** as it allows the programmer to **think** about the **structure** of the program **before** they begin to **create** it.

The most common way to design the logic of a program is to use a text-based notation known as **Pseudocode**. **Pseudocode** is a cross between a **programming language** and our own **English language**. It makes a program **easier** to **understand** without relying on the use of a programs complex **commands** and **syntax**.

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 eventually, **one line** of **pseudocode** becomes **one line** of **program code**. The design language used for the refinements is called **HAGGIS** and is a requirement of the SQA.

Here is the program pseudocode for the first program you will create. The program will take in a message and display that message on the screen in a loop five times. Study the **pseudocode** very closely to understand what is going on:

HAGGIS Design Language

Stepwise Design *(the main steps of the program)*:

1. Setup Variables
2. Get Message
3. Display Message Five Times

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

1. Setup Variables
1.1 **SET** My_Message **TO** String “”

2. Get Message
2.1 **RECEIVE** My_Message **FROM (String) KEYBOARD**

3. Display Message Five Times
3.1 **REPEAT** with loop = 1 to 5
3.2 **SEND** My_Message **TO DISPLAY**
3.3 **END REPEAT**

Stepwise Refinement:
The main steps are broken down further (refined). We use 3.1, 3.2, 3.3, etc.

Notice that the pseudocode looks **more** like our own language rather than that of the programs.

LiveCode

LiveCode is a modern programming environment that has been created by an Edinburgh-based company called Runtime Revolution, **www.runrev.com**.

LiveCode is advertised as being a **very high level language** and is considered to be **even closer** to the way we speak and write as opposed to the sometimes **complex commands** and **syntax** used in other high-level programming environments.

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.

LiveCode is an **event-driven programming language** which 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 very portable which means that it can run on a **variety** of operating system **platforms**. This includes a **PC** running Windows XP, Vista, Windows 7 and 8 or Linux as well as on a **Mac** running OS X.

At least **400MB** of **hard disk** space and **256MB** of **RAM** is required in order for the programming language to run.

The LiveCode programming environment has already been **installed** in the **Applications** folder (mac) or **Program Files** (Windows PC).

IMPORTANT

You will need to **copy** the **LiveCode Programming Tasks** from the **National 4 LiveCode Programming** area of Glow to **your own programming** folder within your user folder. This folder contains the National 4 LiveCode stacks for all 8 tasks that you will do during this programming unit.

N4 > Software Design & Development > N4 LiveCode Programming

LiveCode
Programming Tasks

Getting Started

Task 1: Display My Message (Your First Program!)

Specification

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

The user will be prompted for their **message** and it will be displayed **five times**.

In order to display the message five times, a simple **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 solve the problem.

Here is the **analysis** for the **Display My Message** Program:

I have been asked to create a simple program which asks for a message and then displays the message on the screen 5 times. I will carry out the following **three** steps:

Step 1: I will setup the required variable.
Step 2: I will prompt the user for the message they require.
Step 3: The users message will be displayed in an Output box five 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 and helps the programmer think about the program to be created in a step-by-step manner.

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

HAGGIS Design Language

Stepwise Design *(the main steps of the program)*:
1. Setup Variables
2. Get Message
3. Display Message Five Times

Stepwise Refinement *(breaking down the main steps into smaller steps)*:
1. Setup Variables
1.1 **SET** My_Message **TO** String “”

2. Get Message
2.1 **RECEIVE** My_Message **FROM (String) KEYBOARD**

3. Display Message Five Times
3.1 **REPEAT** with loop = 1 to 5
3.2 **SEND** My_Message **TO DISPLAY**
3.3 **END REPEAT**

Getting Started

Task 1: Display My Message (Your FIRST EVER Program!)

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.

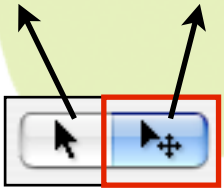
Your program script will be placed into this button

Open the "**Display My Message**" program stack. It can be found in:

N4 LiveCode Programming > 1_Display My Message.livecode

Follow these simple instructions below in order to a produce the script for the "Display My Message" button.

Run Mode **Edit Mode**



Once the program has been opened, select the "**Edit Mode**" to **edit** the script for the "**Display My Message**" button.

Select the "**Display My Message**" button to edit the object's script.

Select the "**Code**" button at the top left of the toolbar.

You will now enter the lines of program code on the next page **very carefully**.

Task 1: Display My Message (Your FIRST EVER Program!)

Implementation (continued)

When you edit the "**Display My Message**" button, you see the area where the program script is to be created. The script consists of lines of program code that will be executed when the button is pressed using the mouse.

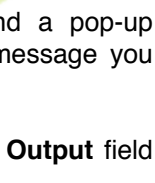
Your task is to add the program code shown below carefully. Watch your spelling as **all** your code will need to be **correct** in order for your program to run successfully!

```
// Setup the variable to be used in this event
Global My_Message

On mouseUp
  // A list of the steps that will occur when the button has been pressed
  Setup_Variables
  Get_Message
  Display_Message_Five_Times
End mouseUp

On Setup_Variables
  // Setup the variable as a string data type
  Put "" into My_Message
End Setup_Variables

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_Five_Times
  // Display users message on the screen 5 times using a loop
  REPEAT with loop = 1 to 5
    Put My_Message & Return after field "Output"
  END REPEAT
End Display_Message_Five_Times
```

Testing

Test that your program produces the correct Output by running it. If the program works correctly, it will display the message you keyed in **five** times.

To run a LiveCode program, go to the **File** menu and **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** field as shown on the right.

Save your program as 1_Display My Message.livecode

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 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 complete this task by carrying out the following **three** steps:

Step 1: Setup the variables required to store the party details.

Step 2: Display a message asking for the user to enter the title of their party. Once the user has entered this, I will then display a 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.

Step 3: 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. Setup Variables
2. Get Invitation Details
3. Display Invitation Details

Stepwise Refinement *(breaking down the main steps into smaller steps)*:
1. Setup Variables
1.1 **SET** Invitation_Title **TO** String ""
1.2 **SET** Invitation_Location **TO** String ""
1.3 **SET** Invitation_Time **TO** String ""
1.4 **SET** Invitation_Date **TO** String ""

2. Get Invitation Details
2.1 **RECEIVE** Invitation_Title **FROM (String) KEYBOARD**
2.2 **RECEIVE** Invitation_Location **FROM (String) KEYBOARD**
2.3 **RECEIVE** Invitation_Time **FROM (String) KEYBOARD**
2.4 **RECEIVE** invitation_Date **FROM (String) KEYBOARD**

3. Display Invitation Details
3.1 **SEND** ["You're invited to "] & Invitation_Title **TO DISPLAY**
3.2 **SEND** ["The location is "] & Invitation_Location **TO DISPLAY**
3.3 **SEND** ["The time is "] & Invitation_Time **TO DISPLAY**
3.4 **SEND** ["The date is "] & Invitation_Date **TO DISPLAY**
3.5 **SEND** ["Don't be late!!"] **TO DISPLAY**
3.6 **SEND** a blank line **TO DISPLAY**

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**" stack. It can be found in:

N4 LiveCode Programming > 2_Party Invitation.livecode

Task 2: Party Invitation Program

Implementation (continued)

Click on the **"Edit Mode"** tool to **edit** the script for the Invitation program.

Select the "**Create Invitation**" button to **edit** the **object script** using the button on the top toolbar.

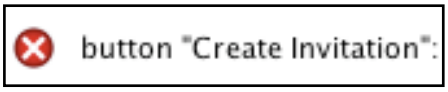
Your task is to add the program code shown below carefully. Watch your spelling as **all** your code will need to be **correct** in order for your program to run successfully!

```
//  Setup the variables to be used in this event.
Global Party_Title, Party_Location, Party_Time, Party_Date

On mouseUp
  //  A list of the steps that will occur when the button has been pressed.
  Setup_Variables
  Get_Invitation_Details
  Display_Invitation_Details
End mouseUp

On Setup_Variables
  //  Setup the variables as string data types.
  Put "" into Party_Title
  Put "" into Party_Location
  Put "" into Party_Time
  Put "" into Party_Date
End Setup_Variables

On Get_Invitation_Details
  //  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, 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
  //  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"
  Put Return after field "Output"
End Display_Invitation_Details
```

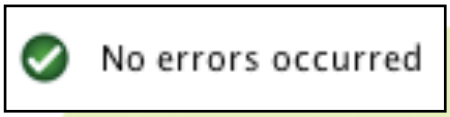
Task 2: Party Invitation Program

Testing

Once your program is complete, check that it works correctly by clicking on the '**Apply**' button.



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



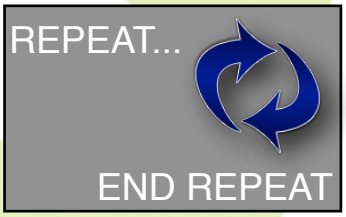
When you have no errors, you are ready to test that your application program works correctly by running it.

Run Tool Edit Tool

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.

Task 2: Party Invitation Extension - 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 setup a **fixed loop** to produce the party ticket 4 times as well as **print** the number of the party ticket (loop). Test that your program works by ensuring that it produces four tickets with the ticket number.




```
On Display_Invitation_Details
  //  Display the details of the party invitation in the Output field.
  REPEAT with loop = 1 to 4
  Put  "This is ticket number: " & loop & 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 as 2_Party Invitation.livecode

What are Variables?

PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE

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 whilst a program is **running**. We give these boxes names which suggest or give us a clue as to what data is being held in the box.

Variable_Name

Data to be stored

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

Party_Title

Steve's 30th Birthday Party

Party_Location

The Balmoral Hotel

Party_Time

6pm until 1am

Party_Date

10th September 2014

Variable names cannot contain any spaces and must **not** be a **reserved** command in LiveCode. 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'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"
```

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 **put** in an **Output field** as shown below:

```
Put "You're invited to " & Party_Title & Return after field "Output"
```

So, what have we learned so far?

PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE PAUSE

The LiveCode program area has three areas:

1. The **variable** list - lists all variables used in the program
2. The **event** - this is a list of all subroutines which are run in order when the button is clicked on by the user.
3. The **subroutines** - 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:

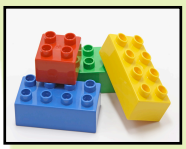
```
Ask "Please enter the title of your party, for example, Easter Chicks Party"
```

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:

```
//  Print out the details of the party ticket
```



On and **End** are used to **begin** and **end** a **subroutine** within an **event**. For example:

```
On Display_Invitation_Details
   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"
   Put Return after field "Output"
End Display_Invitation_Details
```

We know that one way to get one or more lines of code to repeat is a **loop**.

A **Repeat/End Repeat** loop can be used to repeat as many times as we wish.

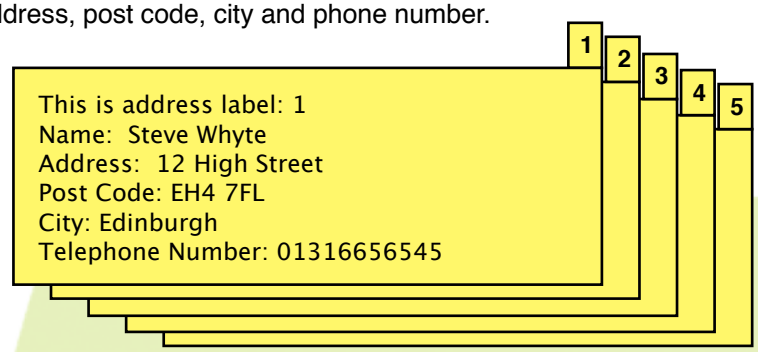
```
REPEAT with loop = 1 to 4
   Put "Hello!" & Return after field "Output"
END REPEAT
```

Well Done! You are <u>now</u> a computer programmer!

Task 3: Address Labels Program

Specification

A program is required in order to produce **5 address labels**. The details to be entered and displayed in a **label** are the contact name, address, post code, city and phone number.



Analysis

I will do this task by dividing it up into **three** steps:
Step 1: I will setup the variables to be used in this program.
Step 2: I will display a message on the screen prompting 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.
Step 3: All details entered will then be displayed 5 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. Setup Variables
2. Get Contact Details
3. Display Contact Details

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

1. Setup Variables
1.1 **SET** Contact_Name **TO** String ""
1.2 **SET** Contact_Address **TO** String ""
1.3 **SET** Contact_Post_Code **TO** String ""
1.4 **SET** Contact_City **TO** String ""
1.5 **SET** Contact_Phone **TO** String ""

HAGGIS Design Language

2. Get Contact Details
2.1 **RECEIVE** Contact_Name **FROM (String) KEYBOARD**
2.2 **RECEIVE** Contact_Address **FROM (String) KEYBOARD**
2.3 **RECEIVE** Contact_Post_Code **FROM (String) KEYBOARD**
2.4 **RECEIVE** Contact_City **FROM (String) KEYBOARD**
2.5 **RECEIVE** Contact_Phone **FROM (String) KEYBOARD**

3. Display Contact Details
3.1 **REPEAT** with loop = 1 TO 5
3.2 **SEND** ["Name: "] & Contact_Name **TO DISPLAY**
2.3 **SEND** ["Address: "] & Contact_Address **TO DISPLAY**
2.4 **SEND** ["Post Code: "] & Contact_Post_Code **TO DISPLAY**
2.5 **SEND** ["City: "] & Contact_City **TO DISPLAY**
2.6 **SEND** ["Telephone Number: "] & Contact_Phone **TO DISPLAY**
3.7 **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**" stack. It can be found in:

N4 LiveCode Programming > 3_Address Labels.livecode

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

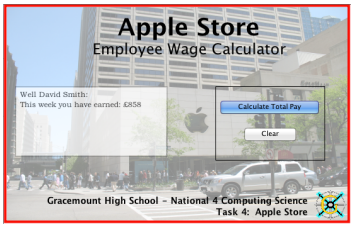
Testing

Test that your program works correctly by producing 5 address labels in the **Output** field.

Save your program as 3_Address Labels.livecode

Task 4: Apple Store

Specification



A program is required to ask for an employee name and work out the amount of money that person 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 week and then display this information on the screen along with a suitable message.

Analysis

I have been asked to create a program to get an employees name and then work out the amount of money they get over the course of a week depending on their hourly wage and amount of hours worked. I will complete this task in **four** steps:

Step 1: Setup the variables to be used in this program
Step 2: Prompt the user for the employees 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 a different variable.
Step 3: In order to calculate the total pay, I will multiply the hours variable by the rate and place this into the variable total_pay.
Step 4: 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 **Apple Store Program**:

Stepwise Design *(the main steps of the program)*:
1. Setup Variables
2. Get Data
3. Calculate Total Pay
4. Display Total Pay

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

1. Setup Variables
1.1 **SET** Name_of_Person **TO** String ""
1.2 **SET** Hours **TO** Real 0.00
1.3 **SET** Rate **TO** Real 0.00
1.4 **SET** Total_Pay **TO** Real 0.00

HAGGIS Design Language

2. Get Data
2.1 **RECEIVE** Name_of_Person **FROM (String) KEYBOARD**
2.2 **RECEIVE** Hours **FROM (Real) KEYBOARD**
2.3 **RECEIVE** Rate **FROM (Real) KEYBOARD**

3. Calculate Total Pay
3.1 **SET** Total_Pay **TO** Hours * Rate

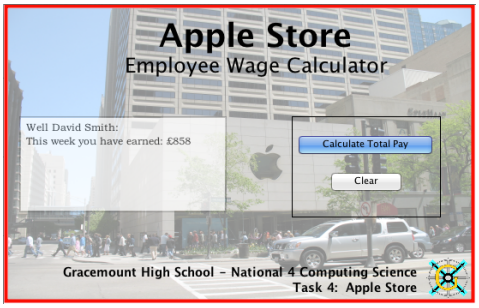
4. Display Total Pay
4.1 **SEND** ["Well "] & Name_of_Person **TO DISPLAY**
4.2 **SEND** ["This week you have earned "] & Total_Pay **TO DISPLAY**

Task 4: Apple Store

Implementation

Open the "**Apple Store**" stack. It can be found in:

N4 LiveCode Programming > 4_Apple Store.livecode



Implementation (continued)

Assign the following code to the "**Calculate Total Pay**" button:


```
// Setup the global variables to be used in this event.
Global Name_of_Person, Hours, Rate, Total_Pay

On mouseUp
  Setup_Variables
  Get_Data
  Calculate_Total_Pay
  Display_Total_Pay
End mouseUp

On Setup_Variables
  // Setup the variables with their different data types.
  Put "" into Name_of_Person
  Put 0.00 into Hours
  Put 0.00 into Rate
  Put 0.00 into Total_Pay
End Setup_Variables

On Get_Data
  // 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_Data

On Calculate_Total_Pay
  // Multiply the hours by the rate. The answer will be placed in to total_pay.
  Put Hours * Rate into Total_Pay
End Calculate_Total_Pay

On Display_Total_Pay
  // Display the users name and total pay in the field "Output".
  Put "Well " & Name_of_Person & ":" & Return after field "Output"
  Put "This week you have earned: £" & Total_Pay & Return after field "Output"
End Display_Total_Pay
```

Task 4: Apple Store

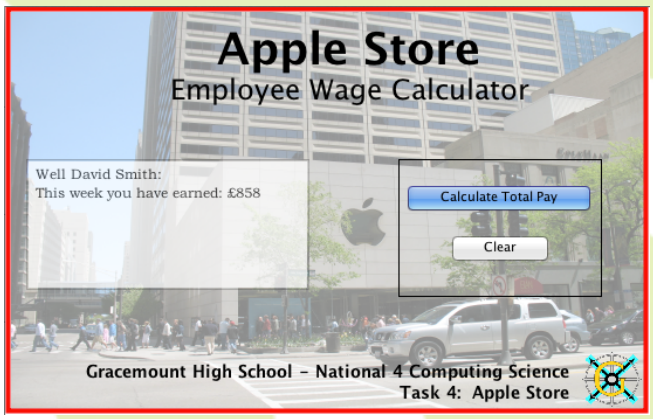
Testing

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.



Name	Hours Worked	Hourly Rate	Calculated Total Pay	Programs Total Pay
Susan Wright	12	£8.00	£96.00	£96.00
Kevin Robertson	30	£10.00	£300.00	£300.00
Allan Drain	39	£20.00	£780.00	£780.00

Save your program as 4_Apple Store.livecode

Task 5: Edinburgh Bowling Club

Specification

An Edinburgh Bowling Club would like a simple program to work out if their members qualify for a special bowling tournament that takes place on an annual basis.

The program should take in the players **name** and the **scores** the player obtained in **four** games.

If the **average score** of their four games is **greater** or **equal** to the **qualifying score** which is set at **100**, then they qualify to compete in the bowling tournament.

If their **average score** is **less than 100**, then they are asked to train harder and try again for the competition next year.

Analysis

The **first** step is to setup the variables used in the program.

The **second** step is to prompt the user for their name and take in the scores from each of their four games.

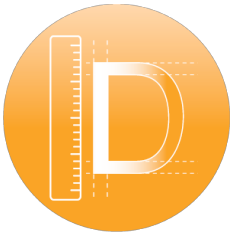
The **third** step is to calculate the average score of the four games by adding up the scores from each game and dividing by four.

The **fourth and final** step involves making a decision as to whether the player qualifies for the bowling tournament by using an IF statement to compare the players average score with the qualifying score and displaying a suitable message.

The Design for the above program is on the next page

Task 5: Edinburgh Bowling Club

Design



Here is the **design** showing the steps for the **Edinburgh Bowling Club**:

Stepwise Design *(the main steps of the program)*:
1. Setup Variables
2. Get Player Data
3. Calculate Average Score
4. Decide If Qualified

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

1. Setup Variables
1.1 **SET** Players_Name **TO** String “”
1.2 **SET** Game1 **TO** Integer 0
1.3 **SET** Game2 **TO** Integer 0
1.4 **SET** Game3 **TO** Integer 0
1.5 **SET** Game4 **TO** Integer 0
1.6 **SET** Average_Score **TO** Real 0.00
1.7 **SET** Qualifying_Score **TO** Integer 100

2. Get Player Data
2.1 **RECEIVE** players_name **FROM (String) KEYBOARD**
2.2 **RECEIVE** Game1 **FROM (Integer) KEYBOARD**
2.3 **RECEIVE** Game2 **FROM (Integer) KEYBOARD**
2.4 **RECEIVE** Game3 **FROM (Integer) KEYBOARD**
2.5 **RECEIVE** Game4 **FROM (Integer) KEYBOARD**

3. Calculate Average Score
3.1 **SET** Average_Score **TO** (Game1 + Game2 + Game3 + Game4) / 4

4. Decide If Qualified
4.1 **SEND** [“Here is the decision for player named ”] & Players_Name **TO DISPLAY**

4.2 **IF** Average_Score >= Qualifying_Score **THEN**
4.3 **SEND** [“You’ve qualified as your average score was ”] & Average_Score **TO DISPLAY**
4.4 **ELSE**
4.5 **SEND** [“You have not qualified as your average was ”] & Average_Score **TO DISPLAY**
4.6 **END IF**

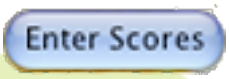
Task 5: Edinburgh Bowling Club

Implementation

Open the "**Edinburgh Bowling Club**" stack. It can be found in:

N4 LiveCode Programming > 5_Edinburgh Bowling Club.livecode

Implementation



Assign the following code to the "**Enter Scores**" button:

```
// Setup the global variables to be used in this event.
Global Players_Name, Game1, Game2, Game3, Game4, Average_Score, Qualifying_Score

On mouseUp
  Setup_Variables
  Get_Player_Data
  Calculate_Average_Score
  Decide_If_Qualified
End mouseUp

On Setup_Variables
  // Setup the variables with their different data types.
  Put "" into Players_Name
  Put 0 into Game1
  Put 0 into Game2
  Put 0 into Game3
  Put 0 into Game4
  Put 0.00 into Average_Score
  Put 100 into Qualifying_Score
End Setup_Variables

On Get_Player_Data
  // Get the players name and scores for each of their four games.
  Ask "Please enter the players name"
  Put it into Players_Name
  Ask "Please enter score for game 1"
  Put it into Game1
  Ask "Please enter score for game 2"
  Put it into Game2
  Ask "Please enter score for game 3"
  Put it into Game3
  Ask "Please enter score for game 4"
  Put it into Game4
End Get_Player_Data

On Calculate_Average_Score
  // Find the average of the four game by adding all them up and then dividing the total by 4.
  Put (Game1 + Game2 + Game3 + Game4) /4 into Average_Score
End Calculate_Average_Score

On Decide_If_Qualified
  // Display a message with the players name.
  Put "Here is the decision for player named " & Players_Name & Return after field "Output"
  // Use an IF statement to determine if the player qualifies for the bowling tournament.
  IF Average_Score >= Qualifying_Score THEN
    Put "You've qualified as your average was " & Average_Score & ". Well done!" after field "Output"
  ELSE
    Put "You've not qualified as your average was " & Average_Score & ". Try harder!" after field "Output"
  END IF
End Decide_If_Qualified
```

Task 5: Edinburgh Bowling Club

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.

Names	Points From Four Games	Calculated Average	Expected Decision	Program Average	Program Decision
Allan Drain	90 120 101 99	102.5	Ready for Tournament	102.5	Ready for Tournament
Steven Whyte	50 80 89 92	77.75	Not Ready for Tournament	77.75	Not Ready for Tournament
Shona Valentine	120 127 123 115	121.25	Ready for Tournament	121.25	Ready for Tournament
Louise Taylor	55 75 68 63	65.25	Not Ready for Tournament	65.25	Not Ready for Tournament

Save your program as 5_Edinburgh Bowling Club.livecode

Task 6: Paintball Games

Specification

A company requires a program to calculate the total cost of its paintball games and work out if a customer gets a free games based on the games that it plays.

The cost of each game ranges from **£10** to **£30** depending on how big the group is and **IF** the total cost of the four games is over £50, the customer is entitled to play in another game of their choice for free.

There are four paintball games that customers can take part in. These are the:

- **Steal the Flag Game**
- **Team Versus Team Game**
- **Capture the Base Game**
- **Strikeout Game**

Analysis

The **first** step is to setup the variables used in the program.

The **second** step is to prompt the user for the price of the four games of paintball. Prices range from £10 to £30.

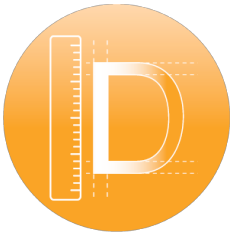
The **third** step is to calculate the total cost of all four games using the prices entered.

The **fourth and final** step involves making a decision as to whether the player is entitled to a free game of paintball using an **IF** statement to decide if the player has spent over £50.

The Design for the above program is on the next page

Task 6: Paintball Games

Design



Here is the **design** showing the steps for the **Edinburgh Bowling Club**:

Stepwise Design *(the main steps of the program)*:

1. Setup Variables
2. Get Game Prices
3. Calculate Total
4. Decide

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

1. Setup Variables
1.1 **SET** Steal_Flag_Game **TO** Real 0.00
1.2 **SET** Team_Game **TO** Real 0.00
1.3 **SET** Capture_Base_Game **TO** Real 0.00
1.4 **SET** Strikeout_Game **TO** Real 0.00
1.5 **SET** Total **TO** Real 0.00

2. Get Game Prices
2.1 **RECEIVE** Steal_Flag_Game **FROM (Real) KEYBOARD**
2.2 **RECEIVE** Team_Game **FROM (Real) KEYBOARD**
2.3 **RECEIVE** Capture_Base_Game **FROM (Real) KEYBOARD**
2.4 **RECEIVE** Strikeout_Game **FROM (Real) KEYBOARD**

3. Calculate Total
3.1 **SET** Total **TO** Steal_Flag_Game + Team_Game + Capture_Base_Game + Strikeout_Game

4. Decide
4.1 **SEND** [“The total cost of all paintball games is £”] & Total **TO DISPLAY**
4.2 **IF** Total > 50 **THEN**
4.3 **SEND** [“You are entitled to an extra paintball game for free!”] **TO DISPLAY**
4.4 **END IF**

Task 6: Paintball Games

Implementation

Open the "**Paintball Games**" stack. It can be found in:

N4 LiveCode Programming > 6_Paintball Games.livecode

Implementation

Assign the following code to the "**Get Prices**" button:

```
// Setup the global variables to be used in this event.
Global Steal_Flag_Game, Team_Game, Capture_Base_Game, Strikeout_Game, Total

On mouseUp
  Setup_Variables
  Get_Game_Prices
  Calculate_Total
  Decide
End mouseUp

On Setup_Variables
  // Setup the variables as real data types.
  Put 0.00 into Steal_Flag_Game
  Put 0.00 into Team_Game
  Put 0.00 into Capture_Base_Game
  Put 0.00 into Strikeout_Game
  Put 0.00 into Total
End Setup_Variables

On Get_Game_Prices
  // Ask the user for the cost of their four different games.
  Ask "Please enter the cost of the Capture The Flag Game: "
  Put it into Steal_Flag_Game
  Ask "Please enter the cost of the Team Versus Team Game: "
  Put it into Team_Game
  Ask "Please enter the cost of the Capture The Base Game: "
  Put it into Capture_Base_Game
  Ask "Please enter the cost of the Strikeout Game: "
  Put it into Strikeout_Game
End Get_Game_Prices

On Calculate_Total
  // Calculate the overall total of each game by adding up the cost of each game.
  Put Steal_Flag_Game + Team_Game + Capture_Base_Game + Strikeout_Game into Total
End Calculate_Total

On Decide
  // Make a decision as to whether the user qualifies for a free game based on the
  // total cost of all games. The total must be over £50 in order to qualify.
  Put "The total cost of all paintball games is £" & Total into line 1 of field "Output"
  IF Total > 50 THEN
    Put "You are entitled to an extra paintball game for free!" into line 2 of field "Output"
  END IF
End Decide
```

Task 6: Paintball Games

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.

Price of Four Games	Calculated Total	Expected Decision	Program Average	Program Decision
£20 £30 £20 £20	£90	Extra Game for Free	£90	Extra Game for Free
£10 £10 £10 £15	£45	No Extra Game for Free	£45	No Extra Game for Free
£20 £5 £10 £20	£55	Extra Game for Free	£55	Extra Game for Free
£20 £20 £5 £5	£50	No Extra Game for Free	£50	No Extra Game for Free

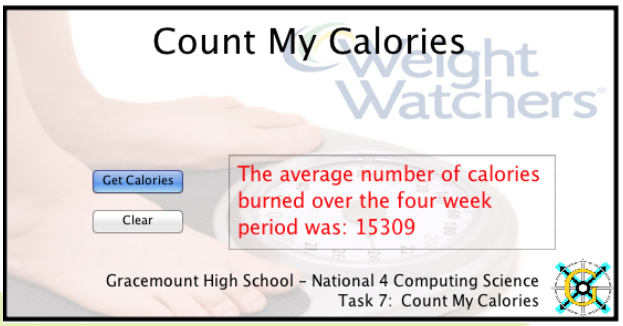
Save your program as 6_Paintball Games.livecode

Task 7: Count My Calories

Specification

A program is required to work out someones average calorie intake over a four week period in order if they are on track to lose weight for Weight Watchers.

The program will take in the users weekly calorie intake over four weeks then will calculate and display the average calorie intake over the course of the month.



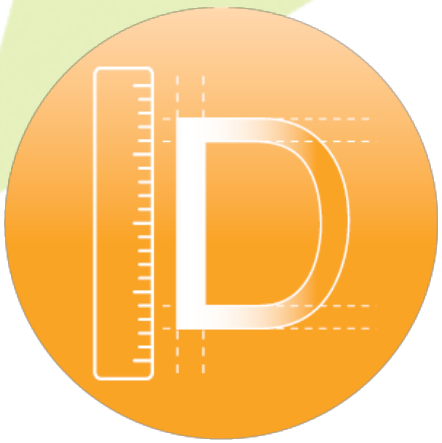
Analysis

The **first** step is to setup the variables used in the program.

The **second** step is to prompt the user for their weekly calorie intake over the course of a month using a REPEAT WITH loop and take in the calories from each of the four weeks and adding this to the total calories variable.

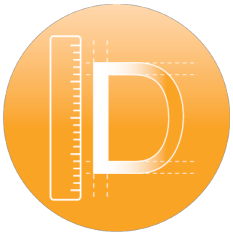
The **third and final** step involves displaying the average number of calories over the four weeks by dividing the total number of calories by the number of weeks (size of the loop).

The Design for the above program is on the next page



Task 7: Count My Calories

Design



Here is the **design** showing the steps for the **Count My Calories**:

Stepwise Design *(the main steps of the program)*:
1. Setup Variables
2. Get Calories
3. Display Average

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

1. Setup Variables
1.1 **SET** Calories **TO** Real 0.00
1.2 **SET** Number_of_Weeks **TO** Integer 4
1.3 **SET** Total_Calories **TO** Real 0.00
1.4 **SET** Four_Week_Average **TO** Real 0.00

2. Get Calories
2.1 **REPEAT** with loop = 1 to Number_of_Weeks
2.2 **RECEIVE** Calories **FROM (Real) KEYBOARD**
2.3 **SET** Total_Calories **TO** Calories + Total_Calories
2.4 **END REPEAT**

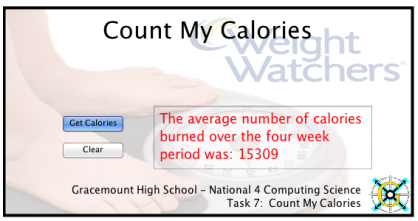
3. Display Average
3.1 **SET** Four_Week_Average **TO** Total_Calories / Number_of_Weeks
3.2 **SEND** [“The average number of calories burned over the 4 week period was: ”] & Four_Week_Average **TO DISPLAY**

Task 7: Count My Calories

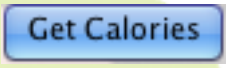
Implementation

Open the "**Count My Calories**" stack. It can be found in:

N4 LiveCode Programming > 7_Count My Calories.livecode



Implementation



Assign the following code to the "**Get Calories**" button:

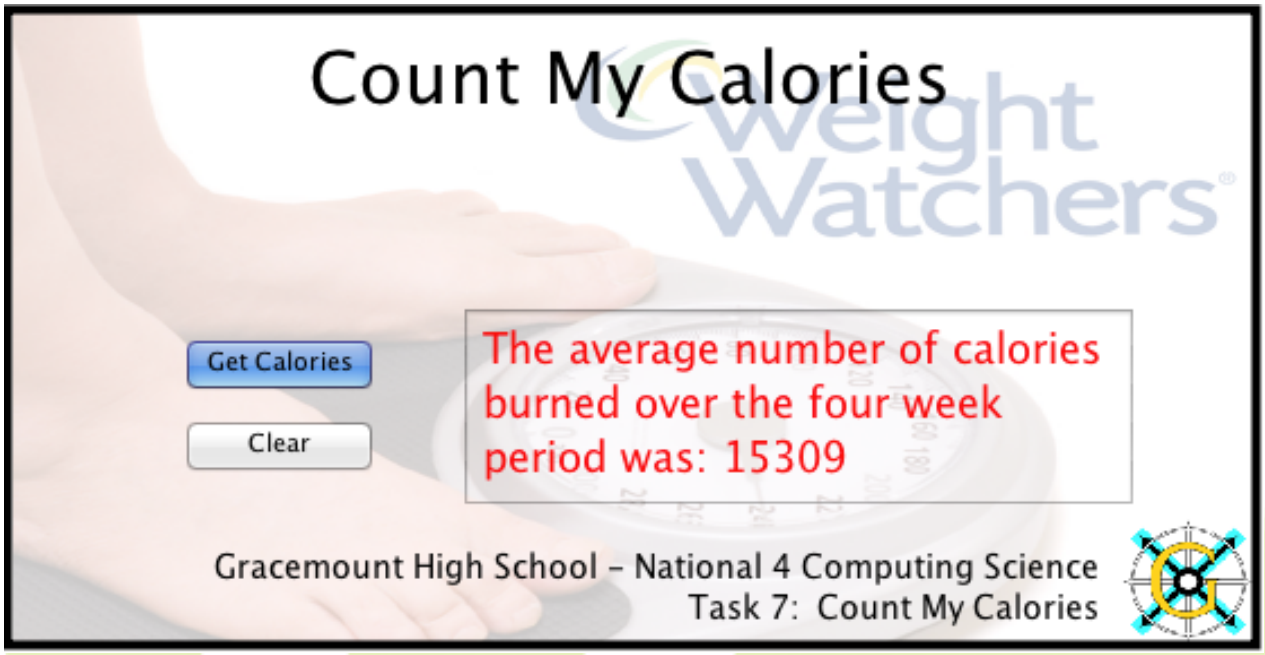
```
// Setup the global variables to be used in this event.
Global Calories, Number_of_Weeks, Total_Calories, Four_Week_Average

On mouseUp
  Setup_Variables
  Get_Calories
  Calculate_and_Display_Four_Week_Average
End mouseUp

On Setup_Variables
  // Setup the variables with their different data types.
  Put 0.0 into Calories
  Put 4 into Number_of_Weeks
  Put 0 into Total_Calories
  Put 0 into Four_Week_Average
End Setup_Variables

On Get_Calories
  // Use a REPEAT loop in order to get the number of calories over the course of
  // the month.
  REPEAT with loop = 1 to Number_of_Weeks
    Ask "Please enter the number of calories burned during week number: " & loop
    Put it into Calories
    Put Calories + Total_Calories into Total_Calories
  END REPEAT
End Get_Calories

On Calculate_and_Display_Four_Week_Average
  // Work out the average and then display a suitable message with the users
  // average monthly calorie intake.
  Put Total_Calories / Number_Of_Weeks into Four_Week_Average
  Put "The average number of calories burned over the four week period was: " &
  Four_Week_Average into line 1 of field "Output"  // On the same line
End Calculate_and_Display_Four_Week_Average
```

Task 7: Count My Calories

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.

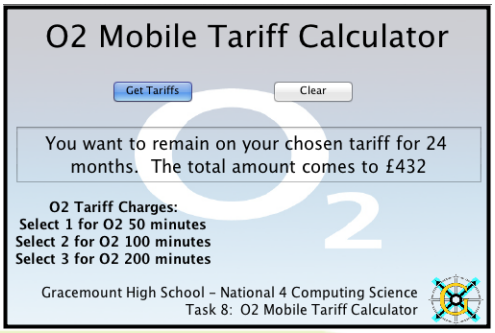


Names	Calories From Four Games	Calculated Average	Program Average
Richard Stewart	14,027 13,050 13,434 12,900	13352.75	13352.75
Paul Main	12,002 11,939 12,325 10,122	11597	11597
Shona King	14,503 15,399 14,492 15,940	15083.5	15083.5
Carol Gray	10,213 10,534 13,655 10,706	11277	11277

Save your program as 7_Count My Calories.livecode

Task 8: O2 Mobile Tariff Calculator

Specification



A program is required to take in a mobile phone contract type. There are **three** contracts to choose from:

- **O2 50 Minutes**
- **O2 100 Minutes**
- **O2 200 Minutes**

Each contract has a different price ranging from cheap to expensive per month. Here are the prices:

Tariff 1: **O2 50 Minutes** for **£11.00** per month
Tariff 2: **O2 100 Minutes** for **£14.50** per month
Tariff 3: **O2 200 Minutes** for **£18.00** per month

The program should ask the user to select a tariff and will then ask the user how many months they would like to remain on that tariff. The program will then calculate the total amount that the user will have to pay back over the course of that period.

Once calculated, the program will display the amount of months the user has chosen and the total amount that they have to pay back.

Analysis

The **first** step is to setup the variables used in the program.

The **second** step is to prompt the user for their choice of contract that they would like to buy using an IF statement.

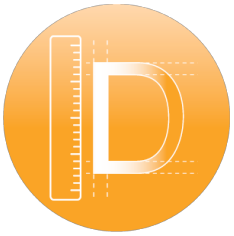
The program will then ask the user for the time in months that they would like to remain on the contract and will carry out a calculation which multiplies the amount of money the contract is per month by the amount of months they wish to take the contract out for.

The **third and final** step involves displaying the length of the contract in months and the total cost the user will have to pay.

The Design for the above program is on the next page

Task 8: O2 Mobile Tariff Calculator

Design



Here is the **design** showing the steps for the **O2 Mobile Tariff Program**:

Stepwise Design *(the main steps of the program)*:
1. Setup Variables
2. Get Tariff Choice
3. Display Overall Cost

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

1. Setup Variables
1.1 **SET** Number_of_Months **TO** Integer 0
1.2 **SET** O2_50 **TO** Real 11.00
1.3 **SET** O2_100 **TO** Real 14.50
1.4 **SET** O2_200 **TO** Real 18.00
1.5 **SET** choice **TO** Integer 0
1.6 **SET** Total_Cost **TO** Real 0.00

2. Get Tariff Choice
2.1 **RECEIVE** Choice **FROM (Integer) KEYBOARD**
2.2 **RECEIVE** Number_of_Months **FROM (Integer) KEYBOARD**

2.3 **IF** choice = 1 **THEN**
2.4 **SET** Total_Cost **TO** O2_50 * Number_of_Months

2.5 **IF** Choice = 2 **THEN**
2.6 **SET** Total_Cost **TO** O2_100 * Number_of_Months

2.7 **IF** Choice = 3 **THEN**
2.8 **SET** Total_Cost **TO** O2_200 * Number_of_Months

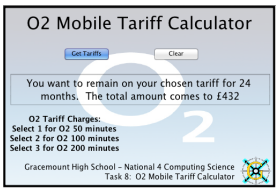
3. Display Overall Cost
3.1 **SEND** [“You want to remain on your chosen tariff for ” & Number_of_Months “The total amount comes to £”] & Total_Cost **TO DISPLAY**

Task 8: O2 Mobile Tariff Calculator

Implementation

Open the "**Mobile Tariff Calculator**" stack. It can be found in:

N4 LiveCode Programming > 8_O2 Mobile Tariff Calculator.livecode



Implementation

Assign the following code to the "**Get Tariffs**" button:

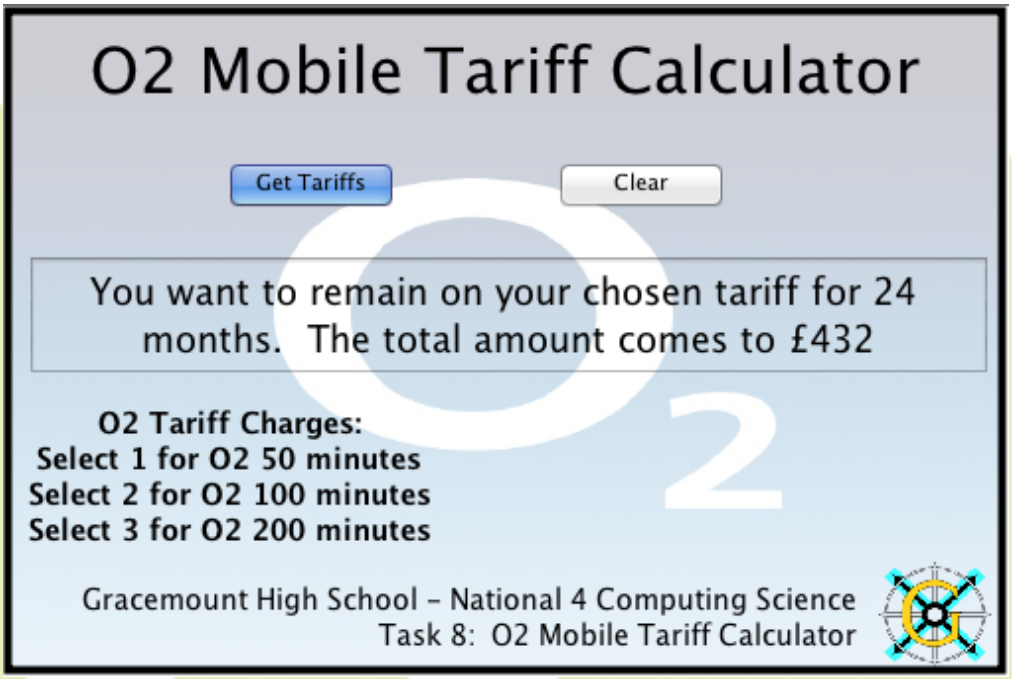
```
//  Setup the global variables to be used in this event.
Global Number_of_Months, O2_50, O2_100, O2_200, Choice, Total_Cost

On mouseUp
  Setup_Variables
  Get_Tariff_Choice
  Display_Overall_Cost
End mouseUp

On Setup_Variables
  //  Setup the variables with the different data types.
  Put 0 into Number_of_Months
  Put 11.00 into O2_50
  Put 14.50 into O2_100
  Put 18.00 into O2_200
  Put 0 into Choice
  Put 0.00 into Total_Cost
End Setup_Variables

On Get_Tariff_Choice
  //  Ask the user for their choice of tariff.
  Ask "Please choose the type of tariff you want.  Enter 1 for O2 50, 2 for O2 100 and 3 for O2 200"
  Put it into Choice

  //  Ask the user for the number of months they would like to remain on the tariff.
  Ask "Please enter the number of months that you want to remain on this tariff"
  Put it into Number_of_Months

  //  If the first tariff is chosen, multiply the cost of the tariff by the number of months the user has
  //  chosen and put the answer into the variable total_cost.
  IF Choice = 1 THEN
    Put O2_50 * Number_of_Months into Total_Cost
  END IF

  //  If the the second tariff is chosen, multiply the cost of the tariff by the number of months the
  //  user has chosen and put the answer into the variable total_cost.
  IF Choice = 2 THEN
    Put O2_100 * Number_of_Months into Total_Cost
  END IF

  //  If the third tariff is chosen, multiply the cost of the tariff by the number of months the user has
  //  chosen and put the answer into the variable total_cost.
  IF Choice = 3 THEN
    Put O2_200 * Number_of_Months into Total_Cost
  END IF
End Get_Tariff_Choice

On Display_Overall_Cost
  //  Display the overall cost of the tariff over the chosen period of months.
  Put "You want to remain on your chosen tariff for " & Number_of_Months & " months.  The total amount
      comes to £" & Total_Cost into line 1 of field "Output"  //  On the same line
End Display_Overall_Cost
```

Task 8: O2 Mobile Tariff Calculator

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.



Contract Name and Cost Per Month	Chosen Number of Months	Calculated Total Cost	Program Average
O2 50 £11.00	24	£264	£264
O2 100 £14.50	18	£261	£261
O2 200 £18.00	12	£216	£216

Save your program as 8_O2 Mobile Tariff Calculator.livecode