**GCSE COMPUTER SCIENCE COURSE WORK**

|  |  |
| --- | --- |
| Decomposition | COMPLETE |
| Planning | INCOMPLETE |
| Programming | NOT STARTED |
| Testing | NOT STARTED |
| Evaluation | NOT STARTED |

**CHECKLIST:**

Phases

There will be 5 phases throughout this project which can all be seen in the checklist above. Below is a breakdown on what I shall do in each phase:

* Decomposition – I shall determine what the problem is and what I need to do to solve it. I shall also state my assumptions for the project and what techniques I shall be using to solve it
* Planning – I shall first use flowcharts to breakdown my program into small chunks that can easily be interpreted to aid my ability to write pseudocode effectively and cleanly (not all over the place)
* Programming – I will transfer my pseudocode into python and then provide tweaks to it to make sure it runs as cleanly as possible and to improve the program further if needed
* Testing – I will test every possible output of my program to ensure that it works and does not crash at any point
* Evaluation – I shall evaluate the effectiveness of my program and check that it works as intended

**Decomposition & Analysis**

Summary:

A local youth club has collected coins throughout the year at various charity events. 6 club members have volunteered to count the coins and have sorted them into the following types of bags:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Coin | Bag Value | Coins in bag | Weight of coin | Weight of bag |
| £2 | £20 | 10 | 12.00g | 120g |
| £1 | £20 | 20 | 8.75g | 175g |
| 50p | £10 | 20 | 8.00g | 160g |
| 20p | £10 | 50 | 5.00g | 250g |
| 10p | £5 | 50 | 6.50g | 325g |
| 5p | £5 | 100 | 3.25g | 325g |
| 2p | £1 | 50 | 7.12g | 356g |
| 1p | £1 | 100 | 3.56g | 356g |

Sub Problems:

* Input for volunteer, coin and weight
* Checking that the weight is correct for the number of coins that should be in the bag – stored as a tuple
* Telling the user what to correct the bag by if it is incorrect
* Keep track of how much money has been collected and allow the user to see this – stored in a csv
* Keep track of the number of bags counted by each volunteer and the accuracy of the volunteers and display it to the user if requested and display it in order of accuracy – stored in CSV
* Save it in a CSV file titled CoinCount.txt

Assumptions:

* The user will not reenter a bag once it has been corrected – I have made this assumption because as soon as the program has told the user what to add, it will add the bag to the program as if it were correct
* All volunteers will be pre-entered into the program – I have done this so there is not lots of different accounts of 1 user which are just slightly different in username as someone entered it in wrong when adding a bag so if there is a new volunteer it will be added using a separate function instead

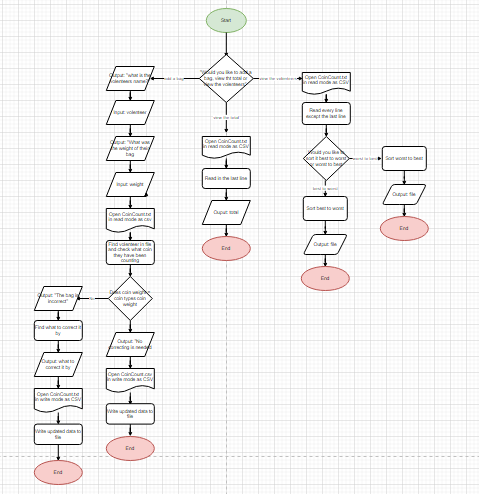
Techniques:

To make this program I will use a variety of techniques; below is a list of some that I shall be using:

* Functions and Procedures (Subroutines)
* If statements
* For and While loops
* Csv reading and writing to/from a file
* Tuples and Arrays
* Try and excepts (to be adding in coding phase) so I can avoid the program crashing

**Planning**

Flowchart:



Psuedocode:

SET coins AS (“1p”,“2p”,“5p”,“10p”,“20p”,“50p”,“£1”,“£2”) AS TUPLE  
SET coinWeight AS (120,175,160,250,325,325,356,356) AS TUPLE  
SET coinAmount AS (10,20,20,50,50,100,50,100) AS TUPLE  
PROCEDURE add()  
BEGIN PROCEDURE  
 SET volunteers AS [] AS ARRAY  
 SET attempts AS [] AS ARRAY  
 SET percents AS [] AS ARRAY  
 SEND “what is the volunteers name?” TO DISPLAY  
 RECEIVE volunteer FROM KEYBOARD AS STRING  
 SEND “what coin did they count” TO DISPLAY  
 RECEIVE coinType FROM KEYBOARD AS STRING  
 SEND “What was the weight of their bag?” TO DISPLAY  
 RECEIVE weight FROM KEYBOARD AS INTEGER  
 SET file AS OPEN CoinCoint.txt IN READ AS CSV  
 FOR row = 1 TO (LENGTH file – 1) DO  
 APPEND volunteers WITH row[0]  
 APPEND attempts WITH row[1]  
 APPEND percents WITH row[2]  
 ENDFOR  
 CLOSE FILE  
 IF volunteer IN volunteers THEN   
 PASS  
 ELSE   
 SEND “that is not a registered volunteer” TO DISPLAY  
 BREAK

ENDIF  
 SET ind AS INDEX volunteer IN volunteers AS INTEGER  
 SET attempt AS attempts[ind] AS INTEGER  
 SET percent AS percents[ind] AS FLOAT  
 SET cWeight AS coinWeight[ind] AS FLOAT  
 IF weight == cWeight THEN  
 SEND “No correcting is needed” TO DISPLAY  
 SET percents[ind] AS ((((percent/100)\*attempt)+1)/attempt+1)\*100 AS FLOAT  
 write(ind,volunteers,coinType,attempts,percents,attempt)  
 ELSE  
 SEND “The bag is incorrect” TO DISPLAY  
 SET percents[ind] AS (((percent/100)\*attempt)/attempt+1)\*100 AS FLOAT  
 fix(ind,volunteers,coinType,attempts,percents,attempt,weight)  
 ENDIF  
END PROCEDURE

PROCEDURE fix(ind,volunteers,coinType,attempts,percents,attempt)  
BEGIN PROCEDURE  
 SET cWeight AS coinWeight[ind]/coinAmount[ind] AS FLOAT  
 IF weight < coinWeight[ind] THEN  
 SET i AS 0 AS INTEGER  
 WHILE weight < coinWeight[ind] DO  
 SET weight AS weight + cWeight AS FLOAT  
 SET i AS i + 1 AS INTEGER  
 ENDWHILE  
 SET i AS “+” + (STRING i) AS STRING  
 ELSEIF weight > coinWeight[ind] THEN  
 SET i AS 0 AS INTEGER  
 WHILE weight < coinWeight[ind] DO  
 SET weight AS weight - cweight AS FLOAT  
 SET i AS i - 1 AS INTEGER  
 ENDWHILE  
 SET i AS i AS STRING  
 ENDIF  
 SEND “you need to ”+i+“coins” TO DISPLAY  
 write(ind,volunteers,coinType,attempts,percents,attempt)  
END PROCEDURE

PROCEDURE write(ind,volunteers,coinType,attempts,percents,attempt)  
BEGIN PROCEDURE  
 SET ind2 AS INDEX coinType IN coins  
 SET attempts[ind] AS attempt + 1 AS INTEGER  
 SET file AS OPEN CoinCount.txt IN READ AS CSV  
 FOR i = (LENGTH file - 1) TO (LENGTH file) DO  
 SET total AS I AS INTEGER  
 ENDFOR  
 CLOSE FILE  
 SET total AS total + coinType\*coinAmount[ind2] AS INTEGER  
 SET file AS OPEN CoinCount.txt IN WRITE AS CSV  
 FOR i IN 1 TO (LENGTH volunteers – 1) DO  
 WRITE volunteers[ind]+“,”+attempts[ind]+“,”+percents[ind]+“/n” TO file  
 ENDFOR  
 WRITE total TO FILE  
END PROCEDURE

PROCEDURE total()  
BEGIN PROCEDURE  
 SET file AS OPEN CoinCoint.txt IN READ AS CSV  
 FOR i = (LENGTH file - 1) TO (LENGTH file) DO  
 SET total AS i AS STRING  
 ENDFOR  
 CLOSE file  
 SEND “there has been a total of £” + total + “collected” TO DISPLAY  
END PROCEDURE

PROCEDURE volunteers()  
BEGIN PROCEDURE  
 SET volunteers AS [] AS ARRAY  
 SET attempts AS [] AS ARRAY  
 SET percents AS [] AS ARRAY  
 SET file AS OPEN CoinCoint.txt IN READ AS CSV  
 FOR row = 1 TO (LENGTH file – 1) DO  
 APPEND volunteers WITH row[0]  
 APPEND attempts WITH row[2]  
 APPEND percents WITH row[3]  
 ENDFOR  
 CLOSE file  
 SET Sorted AS sort(percents) AS ARRAY  
 SET ind AS [] AS ARRAY  
 FOR i = 1 TO LENGTH percents DO  
 APPEND ind WITH INDEX Sorted[i] IN percents AS INTEGER  
 IF i > 1 AND index[i] == index[i-1] AND attempts[ind[i]] > attempts[ind[i-1]] THEN  
 SET a AS ind[i] AS INTEGER  
 SET ind[i] AS ind[i-1] AS INTEGER  
 SET ind[i-1] AS a AS INTEGER  
 ENDIF  
 ENDFOR  
 SEND “would you like to sort it best to worst or worst to best” TO DISPLAY  
 RECEIVE btw FROM KEYBOARD AS STRING  
 IF btw = “best to worst” THEN  
 FOR i = 1 TO LENGTH ind DO   
 SEND i + “. ” + volunteers[ind[i]] + “has had a” + percents[ind[i]] + “%   
 success rate with ” + attempts[ind[i]] + “ attempts” TO DISPLAY  
 ENDFOR  
 ELSEIF btw = “worst to best” THEN  
 FOR i = LENGTH ind TO 1 DO   
 SEND i + “. ” + volunteers[ind[i]] + “has had a” + percents[ind[i]] + “%   
 success rate with ” + attempts[ind[i]] + “ attempts” TO DISPLAY  
 ENDFOR  
 ELSE   
 SEND “That was not a valid input” TO DISPLAY  
 ENDIF  
ENDPROCEDURE

FUNCTION sort(numbers)  
BEGIN FUNCTION  
 IF LENGTH numbers = 1 THEN  
 RETURN numbers  
 ELSE  
 SET mid AS (LENGTH numbers)/2 AS INTEGER  
 SET l AS numbers[:mid] AS ARRAY  
 SET r AS numbers[mid:] AS ARRAY  
 SET l AS sort(l)  
 SET r AS sort(r)  
 SET i AS 0 AS INTEGER  
 SET j AS 0 AS INTEGER  
 SET k AS 0 AS INTEGER  
 WHILE i < LENGTH l AND j < LENGTH r DO  
 IF l[i] > r[j] THEN  
 SET numbers[k] AS l[i] AS INTEGER  
 SET i AS i + 1 AS INTEGER  
 ELSE  
 SET numbers[k] AS r[j] AS INTEGER  
 SET j AS j + 1 AS INTEGER  
 ENDIF  
 SET k AS k + 1 AS INTEGER  
 ENDWHILE  
 WHILE i < LENGTH l DO  
 SET numbers[k] AS l[i] AS INTEGER  
 SET i AS i + 1 AS INTEGER  
 SET k AS k + 1 AS INTEGER   
 ENDWHILE  
 WHILE j < LENGTH r DO  
 SET numbers[k] AS r[j] AS INTEGER  
 SET j AS j + 1 AS INTEGER  
 SET k AS k + 1 AS INTEGER   
 ENDWHILE  
 RETURN numbers  
END FUNCTION

SEND “Would you like to add a bag, view the total or view the volunteers” TO DISPLAY  
RECEIVE option FROM KEYBOARD  
IF option = “add a bag” THEN  
 add()  
ELSEIF option = “view the total” THEN  
 total()  
ELSEIF option = “view the volunteers” THEN  
 volunteers()  
ELSE  
 SEND “That was not a valid input” TO DISPLAY