COMP124 - Assignment 1

Information

Name	Oliver Legg
Student ID:	201244658
Email	O.Legg@student.liverpool.ac.uk

Requirements

In this assignment we are required to solve the following problems in assembly language:

- Get the user input for the module marks of 6 modules
- The user's input must have validation. You mustn't be allowed to enter a number that is <0 or >100
- The input should be stored in an array
- Count all fails and passes **after** the inputs have been taken. (A pass is considered 40% or above)

Approach

I programmed my assignment with the requirements in mind. I knew that I would have to ask multiple questions and I knew that a loop would be good. Therefore I put all the '`"Enter the mark for module x: "`' `char[]`'s into an array (`const char[][]`). Fortunately, all of the `char[]`'s were 31 characters long (31 bytes) so I knew that I could increment a variable in the loop by 31 to keep record of the index. I used a variable called ac1 to hold this data. I also had to consider where I would be putting the inputs of the user, so I created an array called grades with a (length of 6 as I would only be putting 6 module marks in it). Again, I would need to store the index of the array as well, so I created another variable called ac2 (accumulator 2). Which incremented by 4 each loop as each element of the array takes up 4 bytes. Every loop, I put the input of the user into the index of the array then next loop, I would put the input into the next index of the array. I didn't use the '`LOOP`' function because I implemented validation in this. If the user inputted a non-valid input, the loop would still increment by 1 each time and not ask the question again.

After this, I created another loop which checks if the grade is lesser than 40. If it is, it will be sent to a part of the program called fail which increments the fail count by 1. If not, it will increment the win count by 1. Once this is complete I am sent to the finish section of the program where it simply outputs the wins and losses.

When I ran the program, everything worked, except the information only showed for a split second, I used a break point temporarily because I was able to test if things were working. To fix this, I added a user input to the end of the program so that when everything has been displayed, the user must enter something for the program to finish.

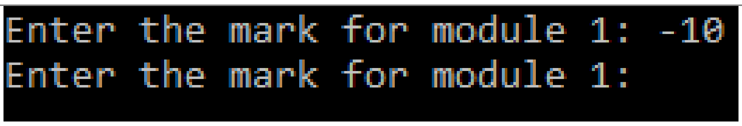
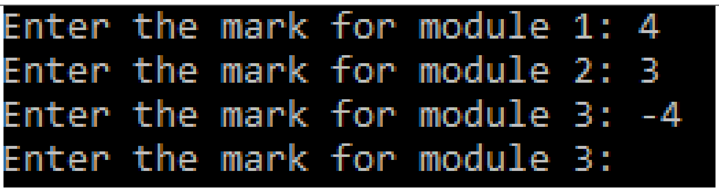
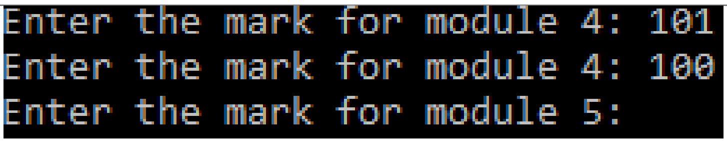
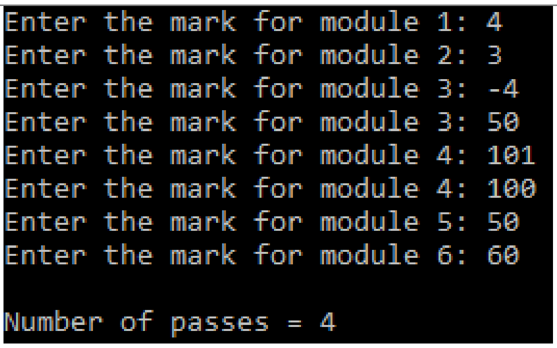
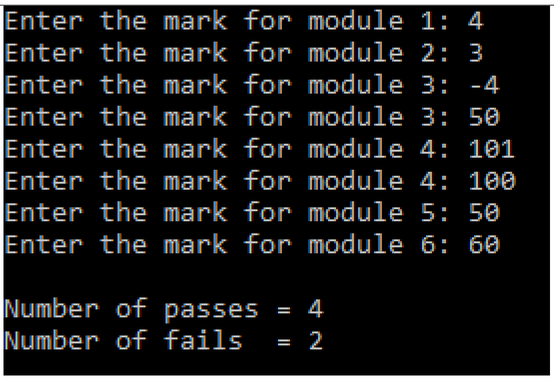
My program solves well because it meets all the requirements and runs efficiently. It is very modular, if you wanted to insert a couple more modules, you can easily do so by changes only a few variables.

Testing

Test

Test number	Description	Expected Results	Actual results	Test data	Remedial Action
1	Input validation 1	Entering a non-valid number should ask you to enter it again	Entering a non-valid number asked me to enter it again	Negative integer (-10)	-
2	Input validation 2	Entering a negative number should ask you to enter it again	The program asked me to enter the number again	Negative integer (-4)	-
3	Input validation 3	Entering a number above 100 should ask me to enter it again	The program asked me to enter the number again	Integer greater than 100 (101)	-
4	Correct passes	The program should display the correct number of passes	The program displayed the correct number of passes	-	-
5	Correct fails	The program should display the correct number of fails	The program displayed the correct number of fails	-	-

Evidence

Test number	**Notes**	**Evidence**
1	Asked me to enter mark for module 1 again because I entered -ve	 `Enter the mark for module 1: -10` `Enter the mark for module 1:`
2	Asked me to enter the mark for module 3 again because I entered -ve	 `Enter the mark for module 1: 4` `Enter the mark for module 2: 3` `Enter the mark for module 3: -4` `Enter the mark for module 3:`
3	Asked me to enter the mark for module 4 again because I entered a number greather than 100	 `Enter the mark for module 4: 101` `Enter the mark for module 4: 100` `Enter the mark for module 5:`
4	Module 3,4,5 and 6 were all passes, therefore this statement is true	 `Enter the mark for module 1: 4` `Enter the mark for module 2: 3` `Enter the mark for module 3: -4` `Enter the mark for module 3: 50` `Enter the mark for module 4: 101` `Enter the mark for module 4: 100` `Enter the mark for module 5: 50` `Enter the mark for module 6: 60` `Number of passes = 4`
5	Module 1 and 2 were both fails which makes the output true.	 `Enter the mark for module 1: 4` `Enter the mark for module 2: 3` `Enter the mark for module 3: -4` `Enter the mark for module 3: 50` `Enter the mark for module 4: 101` `Enter the mark for module 4: 100` `Enter the mark for module 5: 50` `Enter the mark for module 6: 60` `Number of passes = 4` `Number of fails  = 2`