Arrays

Mark Analyser V2.0

Workshop 4 (worth 3% of your final grade)
URL: https://github.com/Seneca-144100/IPC-WS4

In this workshop, you are to upgrade your program in workshop 3 to get the student number of students receiving the marks and as a result of your program print the student numbers and their marks in a list.
Also, you are to make the data entry more user friendly by first printing the placeholder for the data entry and let the user enter the data over the place holders.

An executable demo of this lab is located on matrix at:
`~profname.proflastname/ipc_w4_demo <ENTER>`

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities to:

- Create a more user friendly interactive program.
- Hack your own working code and make it better
- Use arrays to keep all the data entered in the program
- Use parallel arrays to keep the related data together.

SUBMISSION POLICY

Your workshops are divided into two sections; in_lab and at_home.

The "in_lab" section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop. If you do not attend the workshop, you can submit the "in_lab" section along with your "at_home" section (a 20% late deduction will be assessed). The "at_home" portion of the lab is **due the day before your next scheduled workshop**

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible for regularly backing up your work.

IN-LAB: ITEM CLASS (70%)

Download or clone workshop 4 from https://github.com/Seneca-144100/IPC-WS4

Modify your code in workshop 3 part 1 (in_lab) program to do the following:

1- Create two arrays of 40 integers to keep the student numbers and marks. These are parallel arrays. This means that the elements at the same index hold data that are related to the same entity (student number and mark).
2- Modify the title of the application; add V2.0
   - Print:
     >`--=== IPC mark Analyser V2.0 ===--`< and (go to newline)
   - The first character of the title must be on column 8.
3- Modify the prompt to enter the number of **students** rather than marks: (The range of entry is unchanged)
   - Modify the prompt to the following:
     >`Please enter the number of students(between 3 and 40): `<
   - The rest of the entry remains the same
4- Start getting **the student numbers and their marks** one by one. Add students' marks to a total value that you have created for this purpose.
   - Print a title for data entry as follows:
     >`Row   Std No  Mrk`<
     >`--- --------- ---`<
   - Create a loop that repeats to the number of **students** entered.
   - Prompt the user in each repetition by a row number, a place holder for student number and mark entry exactly under corresponding titles
     Use underline characters for place holders:
     (cursor should be standing on the fifth char of the prompt)
     > `  1 _________ ___`<
     To accomplish this first print the place holders >`  _______ __`< then using the carriage return character (\r) move the cursor back to the beginning of the line and finally print the row number using %3d format specifier and a space. This will place the cursor right on the fifth character that is the beginning of student number entry.
   - Then get the student number and mark separated by a space up to the amount entered by the user and place them in the element of the array with the index as the loop counter.
   - (unchanged) Add the value of the mark read to the total. (Make sure the total is initialized to zero) .
5- (unchanged) Divide the total value by the number of marks to find the average of all marks.
6- Close the data entry by printing the following line:
   >`================`< (go to newline)
7- Print all the marks of the students and the result of the division.
   - Print the following message and title:
     >`Marks Entered, printing results:`<
     >`Row   Std No  Mrk`<
     >`--- --------- ---`<
   - Create another loop that repeats to the number of **students** entered.

- In each line print the row, student number and mark separated by space and using these format specifiers respectively: `%03d %09d %3d`
  (%09d, prints an integer in 9 spaces and if the integer is less than 9 characters, it will fill the left spaces with zero)
- Close the list by printing the following line:
  >==========< (go to newline)
- Print >`The average of all marks in this group is XX.X.`<
- Replace "XX.X" with the calculated average.
- There should be one digit after the decimal point.

8- Prompt the end of the program and end the program.
- Print >`Program Ended.`<

Output Example:

Make sure you run the demo at ***~profname.proflastname/ipc_w4_demo*** *to see exactly how the entry should work.*

```
       ---=== IPC mark Analyser V2.0 ===---
Please enter the number of students(between 3 and 40): 2
Invalid number, enter a number between 3 and 40 inclusive: 41
Invalid number, enter a number between 3 and 40 inclusive: 5
Row   Std No  Mrk
--- --------- ---
  1 123456789 45_
  2 456789123 90_
  3 789123456 57_
  4 147258369 89_
  5 014725836 32_
=================
Marks Entered, printing results:
Row   Std No  Mrk
--- --------- ---
001 123456789  45
002 456789123  90
003 789123456  57
004 147258369  89
005 014725836  32
=================
The average of all marks in this group is 62.6.
Program Ended.
```

For submission instructions, see the SUBMISSION section below.

IN_LAB SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above or any information needed.

If not on matrix already, upload your **marks.c** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit ipc_w4_in_lab <ENTER>
```

and follow the instructions.

AT_HOME: TITLE (20%)

Now that the in-lab section of the workshop is done, because of using arrays, all the data are saved (in the arrays) after the entry and we can revisit and process them as many times as needed.

In the previous lab, since we were receiving the data one by one, any new data read had to overwrite the old one. Therefore, we had to do all the processes on the data (i.e. the marks) as we were reading them from the input. Although for our purpose this method worked, processing the data at the moment of entry mixes the logic of data entry and data processing in one step. This can be confusing when the code is revisited later for modifications or possible debugging.

Using arrays we can separate the data entry, the analysis and the presentation (Report and output) of the marks:

We can first loop and get the data and store it in the arrays (See the Data Entry). Now that the in-lab section of the workshop is done, because of using arrays, all the data are saved (in the arrays) after the entry and we can revisit and reprocess them as many times as needed.

1- In the previous lab, since we were receiving the data one by one. Any new data read had to overwrite the old one. Therefore we had to do all the process on the data (i.e. the flowchart attached)
2- Then loop through the data and do the analysis (See Data Analysis flow chart attached)
3- Finally loop through the data and display a comprehensive report of the analysis. (See the Report flowchart attached)

1- Data Entry

After completing the in_lab section, upgrade marks.c to make sure that the marks entered in the array are valid (between 0 and 100 inclusive); use the same logic and error message you used in workshop3 to accomplish this:
>Error: Enter values between 0 and 100 inclusive.\n<

2- Data Analysis

In this loop go through the marks array and find the following stats:

1- Total number of students who passed (greater than or equal to 50) and their average
2- Total number of students who failed (less than 50) and their average
3- Highest mark and its index in the array (new)
4- Lowest mark and its index in the array (new)

To find highest and lowest marks and their indices, you need to define and use two integers to keep track of the indices of the highest and lowest marks in the array.

3- Report (Data presentation)

Start the loop for printing the list of students in rows. When printing the list of the students and their mark, identify the students with highest and lowest mark by printing these prompts in front of their row:
> ***Lowest mark***\n<
> ***Highest mark***\n<

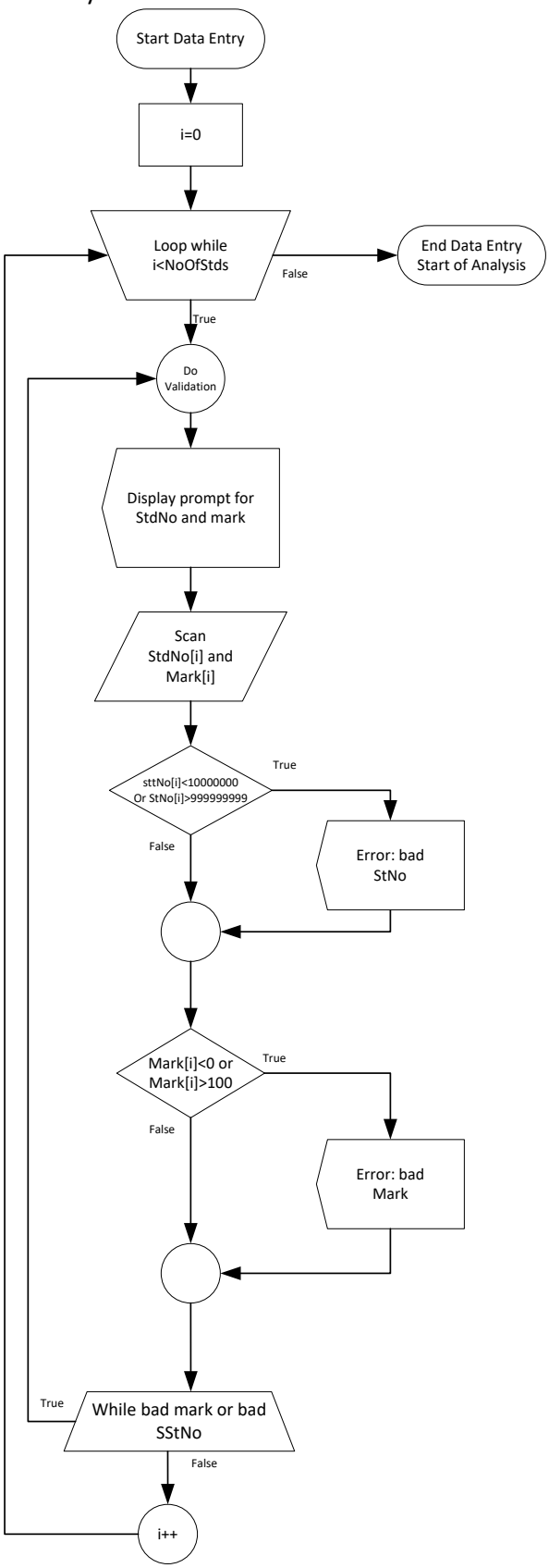
The rest of the report is exactly like workshop3.

Output Example:

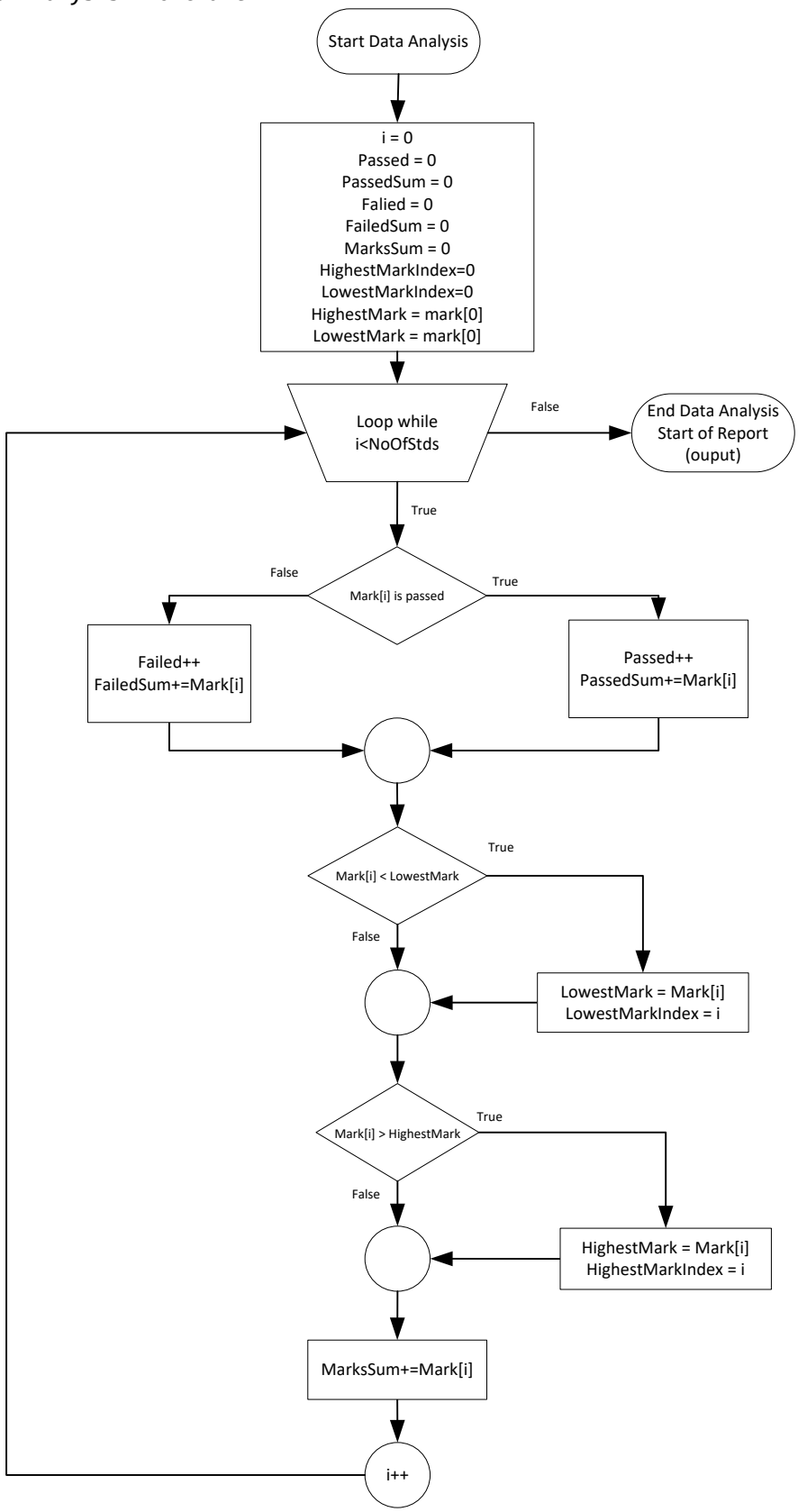
```
       ---=== IPC mark Analyser V2.0 ===---
Please enter the number of students(between 3 and 40): 5
Row   Std No  Mrk
--- --------- ---
  1 123456789 45_
  2 789456123 -1_
Error: Enter mark values between 0 and 100 inclusive.
  2 789456123 101
Error: Enter mark values between 0 and 100 inclusive.
  2 789456123 90_
  3 9999999   57_
Error: Enter student number values between 10000000 and 999999999 inclusive.
  3 1000000000 57
Error: Enter student number values between 10000000 and 999999999 inclusive.
  3 10000000  57_
  4 012345678 89_
  5 147258369 32_
=================
Marks Entered, printing results:
```

```
Row   Std No  Mrk
--- --------- ---
001 123456789  45
002 789456123  90 ***Highest mark***
003 010000000  57
004 012345678  89
005 147258369  32 ***Lowest mark***
================
Total of 3 students passed with an average of 78.7.
Total of 2 students failed with an average of 38.5.
Highest mark in group: 90
Lowest mark in group: 32
The average of all marks in this group is 62.6.
Program Ended.
```

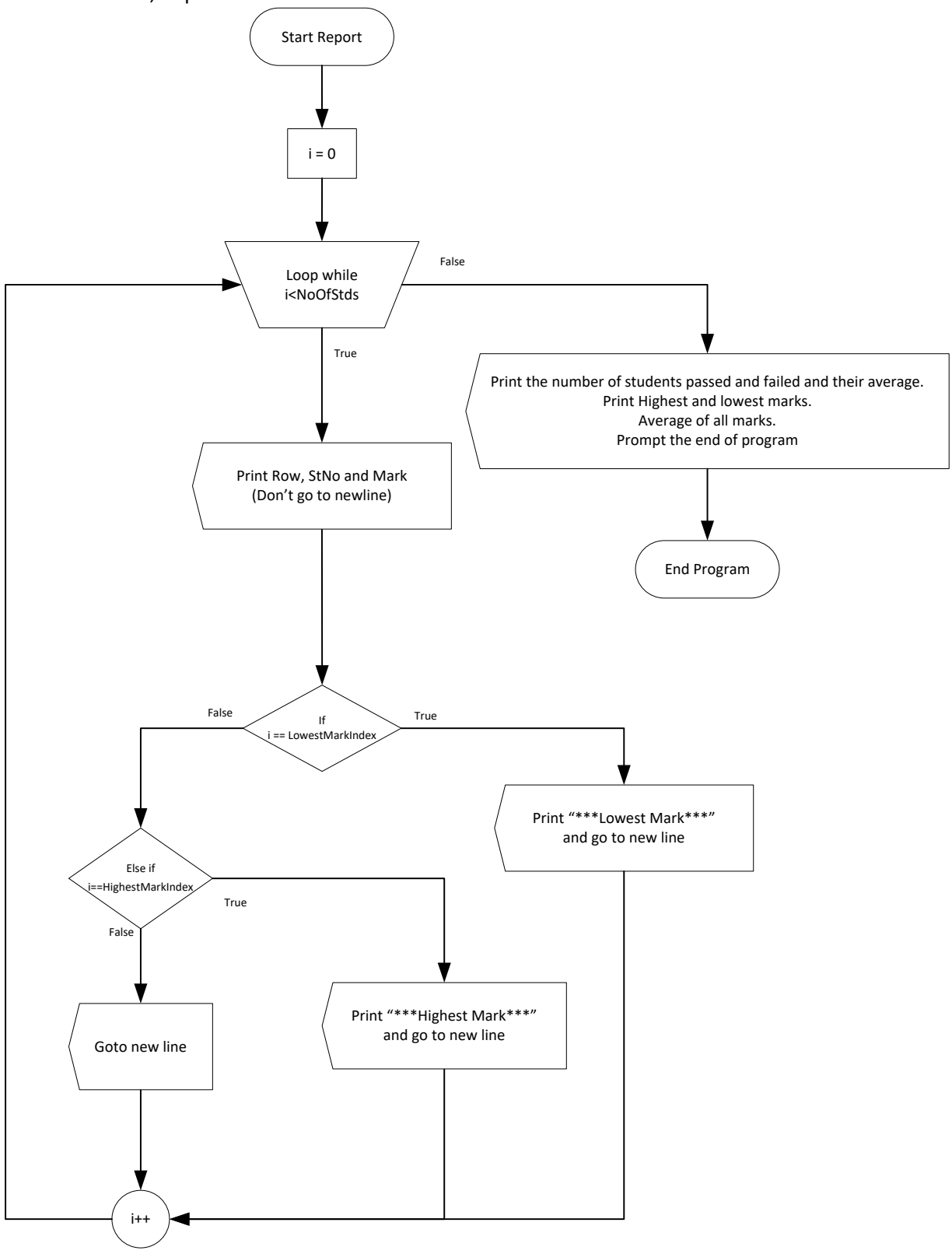
Data Entry Flowchart



Data Analysis Flowchart



Data Presentation, Report



AT-HOME REFLECTION (10%)

Please provide brief answers to the following questions in a text file named **reflect.txt.**

1) In your opinion what are the advantages of dividing the program into Data Entry, Analysis and Report?
2) When initializing an array, what happens if the number of elements initialized is less than the elements in the array?
3) What are parallel arrays and what are they used for?

AT_HOME SUBMISSION:

To test and demonstrate execution of your program, use the same data as the output example above.

If not on matrix already, upload your **marks.c and reflect.txt** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit ipc_w4_at_home <ENTER>
```

and follow the instructions.