CONFIDENTIAL

SCHOOL OF COMPUTING
Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

SKILL BASED TEST (SBT)

SEMESTER I 2018/2019

SUBJECT CODE	**: SCSJ 1013**
SUBJECT NAME	**: PROGRAMMING TECHNIQUE I (PRACTICAL)**
YEAR/COURSE	**: 1 (SCSJ / SCSV / SCSB / SCSR /SCSP)**
TIME	**: 2 HOURS**
DATE/DAY	**: 14th DECEMBER 2018**
VENUE	**: MPK1 – MPK10 (N28)**

INSTRUCTIONS TO THE STUDENTS:

- This test consists of **TWO** questions. You must answer all the questions.
- References to any resources by any means are strictly prohibited.
- You are given **TWO HOURS** to complete the test inclusive of the submission of your program.
- Both of your programs must follow the input and output as shown in the examples.

MATERIAL FOR THE TEST:

- You are provided with **TWO** source code files and **TWO** input files consists of:
  a) A source code file with errors (`marks.cpp`) and input file (`student_data.csv`) for **Question 1.**
  b) A template source code file (`Question2.cpp`) and input file (`input.txt`) for **Question 2**.
- The provided program files should be used as the basis to answer this test.
- Download the files (compressed in ZIP file named `SBT2018.zip`) via UTM's e-learning system**.**

SUBMISSION PROCEDURE:

- Only the source code files (*i.e.* the file with the extension **.cpp**) is required for the submission.
- Submit the source code via the **UTM's e-learning system**.

This question booklet consists of 7 pages inclusive of the cover page.

QUESTION 1 – ERROR DEBUGGING **(35 Marks)**

You developed a program (named "**`marks.cpp`**") to calculate the average marks and display the highest marks for each subject of Computer Science Course. The input file "`student_data.csv`", is a type of Comma Separated Value (CSV) file, consists of name of students and marks for each subject as in Figure 1.1. There are four (4) subjects (Subject 1, Subject 2, Subject 3, and Subject 4) that are represented in column 2 until column 5 respectively as in Figure 1.1. The code runs fine and produces an output as in Figure 1.2 if the program can open the input file successfully, otherwise it displays the error message and exit the program as in Figure 1.3. A friend came to your room and modified the source code (**`marks.cpp`**) as a prank. He modified fourteen (14) lines of code. You are given a C++ program (**`marks.cpp`**) as in Figure 1.4 with errors (syntax errors or/and logical errors, if any) and your task is to **identify** and **fix** the code. Read the data from the provided input (`student_data.csv`), and the program should produce the output as in Figure 1.2 and Figure 1.3 respectively.

```
Tom Riddle,          65,  64,  99,  10
Severus Snape,       97, 100,  60,  75
Harry Potter,        98,  65,  60,  74
Sirius Black,        60,  65,  60,  65
Albus Dumble,        32,  60,  95,  85
```

Figure 1.1: The content of input file, `student_data.csv`

```
Statistics for all subjects :
Subject 1:
        Highest mark = Harry Potter (98)
        Average mark = 70.4
Subject 2:
        Highest mark = Severus Snape (100)
        Average mark = 70.8
Subject 3:
        Highest mark = Tom Riddle (99)
        Average mark = 74.8
Subject 4:
        Highest mark = Albus Dumble (85)
        Average mark = 61.8
```

Figure 1.2: The output if the program is able to open the file.

```
Error opening file student_data.csv
```

Figure 1.3: The output if the program is unable to open the file (input validation).

```
1   #include <oistream>
2   #include <fstream>
3   #include <limits>
4   using namespace std;
5
6   #define FILENAME    "student_data.txt"
7   #define RECORDS     5
8   #define SUBJECTS    4
9   #define MAXSTREAM   numeric_limits<streamsize>::max()
10
11  int main()
12  {
13       //Record for RECORDS number of students
14       char name[RECORDS][100];
15       int marks[RECORDS][SUBJECTS];
16
17       //Statistics
18       float avg[SUBJECTS] = {0};
19       int total[SUBJECTS] = {0};
20       int highest_marks[SUBJECTS] = {0};
21       int highest_name_index[SUBJECTS] = {0};
22
23       //Open file
24       fstream ifile(FILENAME, ios::in)
25
26       //Quit if unable to open a file
27       if (ifile.is_open())
28       {
29       cout << "Error opening file " << "FILENAME" << endl;
30       return -1;
31       }
32
33       //Get data for every student
34       for (int i = 0; i < RECORDS; i++)
35       {
36            //Read name field
37            ifile.get(name[0], 100, ',');
38            ifile.ignore(MAXSTREAM, ','); //Ignore comma
39
40
41
```

```
42  //Read marks for every subject
43            for (int j = 0; j < SUBJECTS-1; j++)
44            {
45                 ifile >> marks[j][i];
46                 ifile.ignore(MAXSTREAM, ',');
47            }
48            ifile >> marks[i][SUBJECTS-1];
49            ifile.ignore(MAXSTREAM, '\n');
50       }
51
52       //Close file
53       ifile.eof();
54
55  //Calculate total marks for each subject and get the
56  //highest marks for each subject
57       for (int i = 0; i < RECORDS; i++)
58       {
59            for (int j = 0; j < SUBJECTS; j++)
60            {
61                 //calculate total marks for each subject
62                 total[j] = marks[i][j];
63
64                 //get the highest marks for each subject
65                 if (marks[i][j] < highest_marks[j])
66                 {
67                      highest_marks[j] += marks[i][j];
68                      highest_name_index[j] = i;
69                 }
70            }
71       }
72
73  //Display statistics for all subjects - the average marks
74  //and highest marks for each subject
75       cout<<"Statistics for all subjects : "<<endl;
76       for (int i = 0; i < SUBJECTS; i++)
77       {
78         //calculate average for each subject
79         avg[i] = total[i] / static_cast<float>(SUBJECTS);
80         cout << "Subject " << i << " : " << endl;
81         cout << "\tHighest mark = "
82            << name[highest_name_index[i]]
83            << " (" << highest_marks[i] << ")" << endl;
84            cout << "\tAverage mark = " << total[i] << endl;
85       }
86       return 0;
    }
```

Figure 1.4: The program (**`marks.cpp`**) with errors.

QUESTION 2 – PROBLEM SOLVING **(65 Marks)**

The Ministry of Health Malaysia is responsible for providing health services to help citizens of Malaysia in sustaining a certain level of health that allows them to lead a productive lifestyle. Assume that you are a programmer working for ministry. You have been asked to write a C++ program to facilitate the ministry to develop a health care system for Body Mass Index (BMI) Calculator module. The module is to measure body fat based on **height in meter** and **weight in kilogram** that applies to adult men and women.

There are seven (7) tasks already listed to complete the program. You are given a partial complete of C++ program (`Question2.cpp`) and an input data file (`input.txt`). Complete the source code according to the tasks given as follows;

TASK 1:

Complete the definition of function `readFile`. This function reads the patient data inputs from the provided input text file named `input.txt`. The patient data consists of patient's name along with their weight in kilogram and height in centimeter. The read data are then stored into arrays accordingly.

Input Validation: Please make sure that the program will only continue reading the file if it is successfully opened, otherwise print the error message and exit the program.

(10 marks)

TASK 2:

Complete the definition of function `convertMeter`, which converts patient's height in centimeter to meter. The formula to convert is as follows:

height (meter) = height (cm) / 100

(5 marks)

TASK 3:

Complete the definition of function **`calculateBMI,`** which calculates the patient's Body Mass Index (BMI) based on **weight in kilogram** and **height in meter** from two arrays correspondingly and puts the results into another array. The formula to calculate BMI is as follows:

BMI = weight (kg) / (height (m) * height(m))

(7 marks)

TASK 4:

Complete the definition of function **`average.`** This function calculates the average value of array elements, which calculates the average weight of patients.

(8 marks)

TASK 5:

Using all functions defined above, write a function call: (a) to read file from the input file, (b) to convert centimeter to meter, (c) to calculate BMI and (d) to calculate average weight of patients.

(9 marks)

TASK 6:

Print the list of patient's names along with their weight in kilogram (kg), height in meter (m), and the calculated BMI to the output file named **`output.txt`** as shown in Figure 2.1.

Note: You should use proper output formatting (align output to the left and number formatting shows two (2) decimal places).

(16 marks)

```
Name                Weight(kg)      Height(m)       BMI

Qisha Sumayyah      70.50           1.40            35.97
Anis Nabila         30.90           1.30            18.28
Adam Zakaria        50.10           1.55            20.85
Mohamad Hafetz      65.30           1.65            23.99
Asma Safiyyah       45.20           1.32            25.94
Ahmad Ali           85.60           1.72            28.93
```

Figure 2.1: Sample output in `output.txt`

TASK 7:

Print the number of patients and the average weight for all patients to the program screen as shown in **Figure 2.2**. ***Note:*** You should use proper output formatting (number formatting for average shows one decimal place). **(5 marks)**

```
Writing to the output file…

Number of patients  : 6
Average weight      : 57.9
```

Figure 2.2: Sample output in program screen.

Table 1: Assessment Criteria

Item	Criteria	Marks
A	The program is able to run and display correct output	4
	Applying proper styles (e.g. indentation, comments)	1
B	Open and close input file	2.5
	Check if the file is successfully open ➔ if fail, display error message and exit the program	3
	Get the name of patients, weight and height from input file and store them into arrays accordingly.	4.5
C	Complete a function to convert centimeter to meter.	5
D	Complete a function to calculate BMI.	7
E	Find the total weight.	4.5
	Find the average weight.	2.5
	Return the average weight.	1
F	Call `readFile` function	2.5
	Call `convertMeter` function	2
	Call `calculateBMI` function	2.5
	Call `average` function	2
G	Open and close output file	2.5
	Print all patient's name, weight, height in meter and the calculated BMI to the output file with proper output formatting.	13.5
H	Print the number of patients and the average weight of patients to the program screen	5
	Total	**65**