Instructions:

- Submit the soft copy of the programs. You don't have to submit the files. Keep all of them in a folder in your home directory in usha macjine.
- **Last date of submission is May 9, 2014, non-extendible.**
- A viva will be conducted after submission.

IMPORTANT : Some more programming problems will be added soon.

Lab1 :

The *wind chill index* (WCI) is calculated from the wind speed v in miles per hour and the temperature t in Fahrenheit. Three formulas are used, depending on the wind speed:

if (0 <= v <= 4) then WCI = t
if (v >=45) then WCI = 1.6t - 55
otherwise, WCI = 91.4 + (91.4 - t)(0.0203v - 0.304$(v)^{1/2}$ - 0.474). Write a program that can calculate the wind chill index.

Answer:
The if-else is suitable for this solution, choosing from three conditional expressions. We need to prompt user for *v* and *t* in order to calculate and show the *wci*. The following is an algorithm for this program using a flow chart.

A sample output:

```
Enter wind speed in mph (-1 to stop): 3.45
Enter temperature in Fahrenheit: 80.25

For wind speed =  3.45 and temperature = 80.25
Wind Chill Index is: 80.25

Enter wind speed in mph (-1 to stop): 20.5
Enter temperature in Fahrenheit: 90.2

For wind speed =  20.50 and temperature = 90.20
Wind Chill Index is: 89.68

Enter wind speed in mph (-1 to stop): 55
Enter temperature in Fahrenheit: 50.7

For wind speed =  55.00 and temperature = 50.70
Wind Chill Index is: 26.12

Enter wind speed in mph (-1 to stop): -1
This program was stopped by you. thank you!
Press any key to continue . . .
```

Lab 2:

Write a program that asks the user to enter any number of integers that are in the range of 0 to 30 inclusive and count how many occurrences of each number are entered. Use a suitable sentinel to signal the end of input. Print out only the numbers (with the number of occurrences) that were entered one or more times. (Note: You must use array in your solution for this problem)

Answer: Uses 3 arrays, one for storing the input, one used for comparison to count the occurrences and another on to store the count of occurrences. Display the content of the third array.

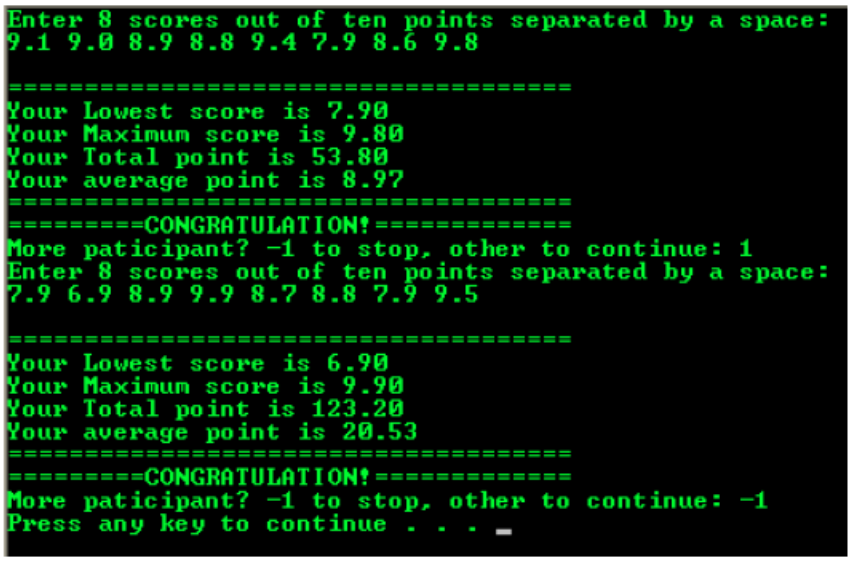
A sample output:

```
Enter integer between 0 and 30 inclusive, other to stop: 3
Enter integer between 0 and 30 inclusive, other to stop: 5
Enter integer between 0 and 30 inclusive, other to stop: 6
Enter integer between 0 and 30 inclusive, other to stop: 7
Enter integer between 0 and 30 inclusive, other to stop: 12
Enter integer between 0 and 30 inclusive, other to stop: 2
Enter integer between 0 and 30 inclusive, other to stop: 3
Enter integer between 0 and 30 inclusive, other to stop: 7
Enter integer between 0 and 30 inclusive, other to stop: 6
Enter integer between 0 and 30 inclusive, other to stop: 5
Enter integer between 0 and 30 inclusive, other to stop: 6
Enter integer between 0 and 30 inclusive, other to stop: 4
Enter integer between 0 and 30 inclusive, other to stop: 10
Enter integer between 0 and 30 inclusive, other to stop: 3
Enter integer between 0 and 30 inclusive, other to stop: 8
Enter integer between 0 and 30 inclusive, other to stop: 9
Enter integer between 0 and 30 inclusive, other to stop: 7
Enter integer between 0 and 30 inclusive, other to stop: 5
Enter integer between 0 and 30 inclusive, other to stop: 3
Enter integer between 0 and 30 inclusive, other to stop: -1
Number          Count
======          =====
2               1
3               4
4               1
5               3
6               3
7               3
8               1
9               1
10              1
12              1
Total user input = 19
Press any key to continue . . .
```

Lab 3:

In a gymnastics or diving competition, each contestant's score is calculated by dropping the lowest and highest scores and then adding the remaining scores. Write a program that allows the user to enter eight judges' scores and then outputs the point received by the contestant. A judge awards point between 1 and 10, with 1 being the lowest and 10 being the highest. For example, if the scores are: 9.2, 9.3, 9.0, 9.9, 9.5, 9.5, 9.6 and 9.8, then the contestant receives a total of 56.9 points. (Note: You must use array in your solution for this problem)

A sample output:



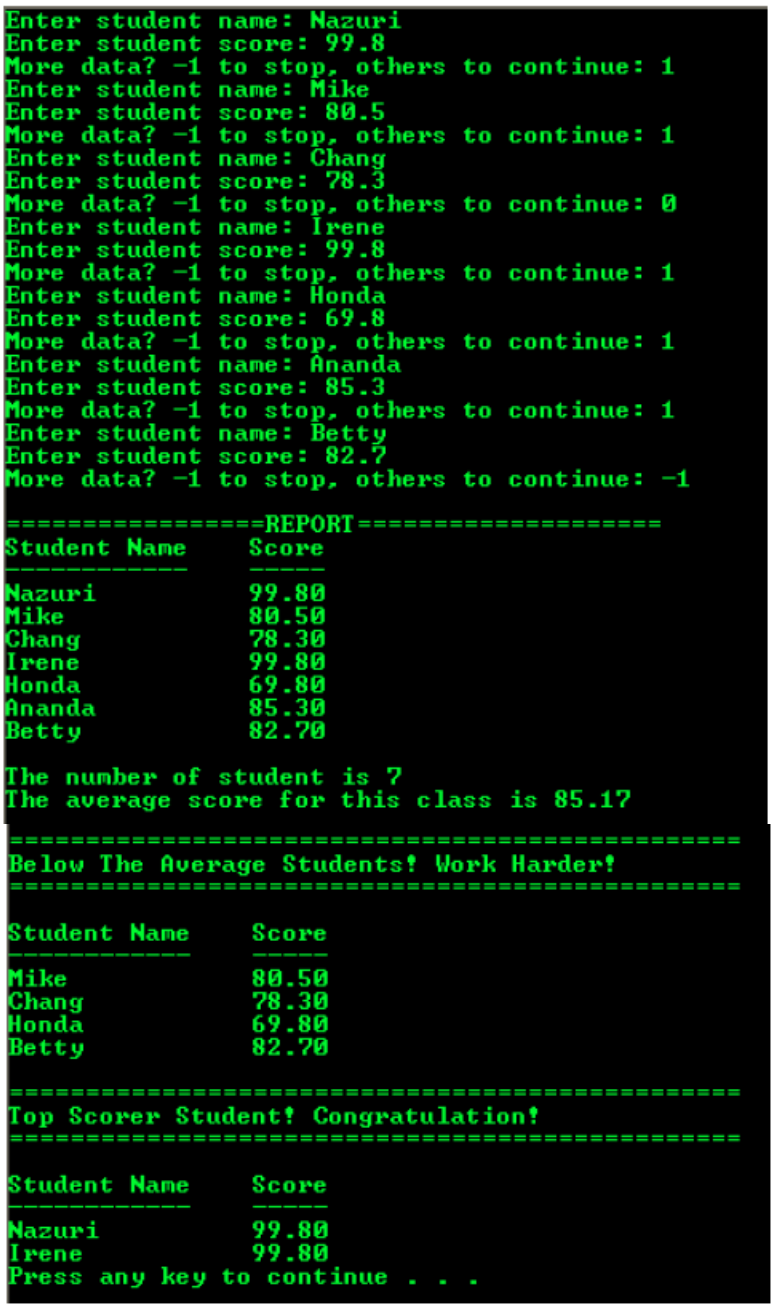
Lab 4:

Write a program that allows the user to enter students' names followed by their test scores and outputs the following information (assume that maximum number of students is 50):

a. The average score.
b. Names of all students whose test scores are below the average, with an appropriate message.
c. Highest test score and the name of all students having the highest score.

Answer: Use 2 arrays to store the student names and scores respectively and then manipulate the array contents.

A sample output:



Lab 5:

A polynomial f(x) is of the form $a_n x^n + a_{n-1} x^{n-1} + .. + a_1 x + a_0$. Here n is the degree of f(x) (n may be zero) and a_n, a_{n-1}, ., a_0 are the coefficients of f(x) with a_0 the constant term of f(x). A number a is a root of f(x) if f(a) = 0. When we replace x by a number b, we are saying we are evaluating f(x) at x = b.

A polynomial of degree 2 is called quadratic polynomial, degree 1, a linear polynomial, degree 0 a constant polynomial (and a polynomial of degree 3 a cubic polynomial).

Polynomials are mathematical functions used very often. You should have encountered many different polynomial operations before such as the sum, difference, and product of two polynomials (done manually).

A polynomial of degree > 1 can be factored if a root a exists or in general two polynomials g(x) and h(x) both of degree > 1 can be found so that f(x) = g(x) h(x). Polynomial factorization and roots are problems beyond the discussion of this lab (a real root can be found for polynomials of real coefficients and odd degree using numerical methods such as Newton's method in, Numerical Methods).

Here we will cover the programming question of how to write a program that computes f(x) for x = b for polynomials of any degree n.

Simple Coding:

For a specific polynomial f(x) of fixed degree, usually f(x) can be coded by simple syntax conversion.

Example 1. $f(x) = 2x^2 + 3.2\ x + 5$; You can convert the mathematical formula to the C programming equivalent formula of 2 * x * x + 3.2 * x + 5 (note 3.2 x must be converted as 3.2 * x, not 3.2x, $2x^2$ must be converted to 2 * x * x, since the format x^2 is not defined in C language).

Now you can write a function that takes a value b, and returns f(b) by using returning the value 2 * x * x + 3.2 * x + 5

Example 2.

$f(x) = x^5 + 3x^4 + 7x + 6$;

You can write and return in the function the value x * x * x * x * x +3 * x * x * x * x + 7 * x + 6. Note how clumsy this function is when you have the 5th power of x. Imagine how awkward this looks if you have a polynomial of degree 8.

Note.

Some students codes x^5 by pow (x, 5) so f(x) will become pow (x, 5) + 3 * pow (x, 4) + 7* x + 6 (pow () in <math.h> library). Although mathematically and programmatically this is a viable approach, it is an overkill since pow (x, 5) is actually exp (x ln 5) and it uses a for loop to implement.

Now consider a general polynomial of degree n $a_nx^n + a_{n-1}\ x^{n-1} + .. + a_1x + a_0$. It will be unthinkable trying to code x * x * x … * x n times if n is big (like 40) or n is unknown.

The polynomial can be evaluated using an array to represent the coefficients and an algorithm called **<u>Horne's method</u>** based on this observation.

$F(x) = ((….((a_nx + a_{n-1}) x + a_{n-2}) x + a_{n-3})x +… + a_1) x + a_0 + a_nx^n + a_{n-1}\ x^{n-1} + .. + a_1x + a_0$

Instead of computing the highest degree term a_nx^n as the product of an multiplied by x * x …x n times (or a_n * pow (x, n)), we calculate first $(a_nx + a_{n-1})\ x = a_nx^2 + a_{n-1}x$, and then $(a_nx + a_{n-1})\ x + a_{n-2})\ x = a_nx^3 + a_{n-1}x^2 + a_{n-2}\ x$, and then $(a_nx + a_{n-1})\ x + a_{n-2})\ x + a_{n-3})x = a_nx^4 + a_{n-1}x^3 + a_{n-2}\ x^2 + a_{n-3}x$ so that we see that gradually we get a_nx, a_nx^2, a_nx^3, a_nx^4, and finally a_nx^n, and $a_{n-1}x$, $a_{n-1}x^2$, $a_{n-1}x^3$ etc until $a_{n-1}x^{n-1}$, and then in a nested way, all the terms with coefficients are calculated and we get f(x) finally.

Your work :

Write a C program to evaluate a polynomial f(x) of degree n for a given value x = b.

Answer :

Step 1. Declare and initialize an array a of n + 1 entries corresponding to the n + 1 coefficients of f(x);

a[0] = a_0, a[1] = a_1, .., a[n] = a_n.

Step 2. Accept a value b from the keyboard

Step 3. Evaluate f(b) in this for loop

```
double term = poly [polyLength-1];
for (int j = polyLength - 2; j >= 0; j --)
      term = term * x + poly [j];
```

Here polyLength is the size of array, which is one plus the degree of the polynomial (a polynomial of degree n has n + 1 coefficients unless this is the zero polynomial).