Assignment V

Submit all the programs separately against each assignment in the Moodle System. Provide the result in a separate output file (named, result_<assgn><no>.txt). Use standard output redirection feature to generate the output file.

Hints. If you run the program with the following command

./a.out >result.txt

Output of your program (generated by printf(.) function) will be written in the file result.txt. You need to provide input from your keyboard, by remembering the sequence of inputs to be given or writing them in a text file in the same sequence.

Otherwise you may use the redirection for the standard input file, such as,

./a.out <input.txt

For the above all your printing by printf(.) function would be displayed on your monitor.

For both reading from a file and writing to a file use the following.

./a.out <input.txt >result.txt

If you execute the program multiple times, you may concatenate the outputs in a single file by using the following redirection command:

./a.out >>result.txt

or

./a.out <input.txt >> result.txt

(a) Consider the following function prototype.

int **check_triangle**(float l1, float l2, float l3);

The function takes three real numbers, l1, l2 and l3 as lengths of straight line segments, and returns 1 if they are capable of forming a triangle, else it returns 0.

Write a C-program to implement the above function and the program reads three lengths and using the above function, prints 'Possible to form a triangle' if they can, else it prints 'Not possible to form a triangle'.

Run the program with the following input data set:

(i) *3, 5, 7*
(ii) *2, 10, 4*
(iii) *1, 2, 3*

(b) Consider the following function prototype.

float **Median3**(float *a*, float *b*, float *c*);

The function takes three real numbers *a*, *b*, and *c* as input and returns their median value.

Write a program which implements **Median3(.)**.The program reads an array of *N* (to be read) values, and computes median values of every consecutive three input array elements (e.g. for array indexes *i-1*, *i*, and *i+1*), and store their median values computed by using **Median3(.)** in another array in the same order (i.e. for numbers at indexes *i-1*, *i*, and *i+1* the median value is stored at the *i* th index of the output array). It assumes all the values beyond the limits of valid array indexes in the input array are zeroes. Finally the program prints both the original array and the processed array (array of median values).

For example given the following input:

N=5

Input array: -3.5 4.7 2.0 -1.0 10

The processed array (output) is:

0 2.0 2.0 2.0 0

Run the program with the following input data set:

(i) N=10

Input array: 25.6 12.7 -4.5 8.9 7.5 6.8 -5.0 1.0 2.5 20.3

(ii) N=8

Input array: -25.6 12.7 4.5 -8.9 7.5 -6.8 -5.0 1.0

(c) Consider the following function prototype.

int **compute_kth_minimum**(int A[ ], int N, int k);

The above functions takes an array A of size N and returns its k-th ($k \leq N$) minimum.

For example, given an array B of 5 integers with values 25, -35, 45, 10, and 0, its 4-th minimum is 25. Hence, **compute_kth_minimum**(B,5,4) returns 25.

Write a C-program which implements the above function and in the main(.) function it reads N (to be read) integers and the value of k. The program returns the sum of first k minima of the array.

For example for the following input:

N=5
The values in the array: 25, -35, 45, 10, 0.
k=3

The program returns (-35+0+10=) -25 as the sum of first 3 minima.

Run the program with the following input data set:

(i) N=10
Input array: 25.6 12.7 -4.5 8.9 7.5 6.8 -5.0 1.0 2.5 20.3
k=7

(ii) N=8
Input array: -25.6 12.7 4.5 -8.9 7.5 -6.8 -5.0 1.0
k=4

N.B. Your program may be tested with other input data set.