Assignment VII

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) Define a data type named **_VECTOR** containing the following information.

dim: An integer denoting the dimension of the vector (assume maximum dimension could be 100).
field: An array of real numbers implementing the vector of dimension of size given in dim.

Write a program which reads an array of M(to be read) vectors of dimension N (to be read), and also print them in the same order. Then it performs a query processing in a loop in the following way.

In each iteration of the loop, it reads a query vector of dimension N and prints the vector from the array which has minimum magnitude of the dot product with the query vector.The query processing terminates and the program ends with a message “Query processing is over”, when the query vector is a zero vector.

For example, if the array contains the following three vector.
(1.5, -4.5, 2.8), (1.6, 0, -4.8), (4.2,1.8,3.7)
For a query vector (1,1,1), it will return (1.5, -4.5, 2.8) as the magnitude of its dot product with (1,1,1) is the minimum among the three. When the query vector is (0,0,0), the loop is terminated and the program ends with a message "Query processing is over".

Run your program with the following input data.

(i) M=4, N=3
(1.5, -4.5, 2.8), (1.6, 0, -4.8), (4.5, -3.5, 1.8), (15.0, 1.0, -4.8)

Query vectors:
(1.2, -3.4, 1.4)
(3.5, 4.6, 0)
(1.45, -3.49, -8.1)
(0,0,0)

(ii) M=5, N=4

(1.5, -4.5, 2.8, -7.6), (1.6, 0, -4.8, 8.3), (4.5, -3.5, 1.8, 4.5), (15.0, 1.0, -4.8, 75), (1.45, -3.49, -8.1, 18.2)

Query vectors:
(1.2, -3.4, 1.4,1.2)
(3.5, 4.6, 0,2.6)
(1.45, -3.49, -8.1,7.4)
(0,0,0,0)

(b) Define a data-type named as **_COMPLEX** which has two member variables of type float for representing real and imaginary parts of a complex number. Write following functions as described below:

(a) **Add_complex_numbers( )**: It adds two complex numbers (passed as arguments) and returns the result of addition.
(b) **Subtract_complex_numbers( )**: It subtracts two complex numbers (passed as arguments) and returns the result of subtraction.
(c) **Multiply_complex_numbers( )**: It multiplies two complex numbers (passed as arguments) and returns the result of multiplication.
(d) **Divide_complex_numbers( )**: It divides two complex numbers (passed as arguments) and returns the result of division.
(e) **Magnitude( )**: Returns the magnitude of a complex number (passed as an argument).
(f) **Phase( )**: Returns the phase of a complex number (passed as an argument).
(g) **Conjugate( )**: Returns the conjugate of a complex number (passed as an argument).
(h) **Negation( )**: Returns the negative value of a complex number (passed as an argument).

Write a main program, which implements a calculator performing all the above operations sequentially over the result of the previous operation (stored in a variable named as **ACCUMULATOR**). At every step, the user will be given the choice of various operations, which may be selected by inputting an integer (such as Add(0), Subtract (1),

etc.). If the operation requires any additional parameters, those are to be also inputted. The result will be displayed after each operation (as it is shown in a calculator). A complex number is displayed / printed in the form of *a + i b*, where *a* is the real part and *b* is the imaginary part.Phase is displayed in radian, and magnitude is displayed as a positive real number. The initial value of **ACCUMULATOR** is zero, and a special operation named as **AC** (All Clear), sets its value to zero. The program terminates with an option for **Exit**, by printing a message '**Exited from the calculator of complex numbers**'.

Provide the output for the following sequence of operations.

(i) Add 3+ i4

(ii) Add 7-i10

(iii) Subtract 110+i23

(iv) Multiply 23-i47

(v) Divide 35+i24

(vi) Conjugate

(viii) Add -35.4+i42.9

(ix) Negation

(x) Phase

(xi) Magnitude

(xii) Multiply 35-i75.4

(xiii) Magnitude

(xiv) Phase

(xv) Add 0+i0

(xvi) AC

(xvii) Exit