HBM803E - Object Oriented Programming Techniques

HOMEWORK 0 (<u>due to Tuesday 10/11/2020 18:00PM</u>)

Questions

1. **(20pt)** Use only **cout<<"*"; cout<<"_";** and **cout<<endl;** statements and algorithmic components (like loops) to produce following outputs on the screen. Your programs should take an integer n which determines number of lines in the figures (for example using "cin>>n;"). All figures worth 4 pts.

   a)

   ```
   *
   **
   ***
   ****
   *****
   ```

   for n=5.

   b)

   ```
   _____*
   ____**
   ___***
   __****
   _*****
   ******
   ```

   for n=6.

   c)

   ```
   __*
   _***
   *****
   ```

   for n=3.

   d)

   ```
   __*
   _***
   *****
   _***
   __*
   ```

   for n=5.

   e)

   ```
   __*
   _*_*
   *___*
   *___*
   _*_*
   __*
   ```

   for n=6.

   Note: when n is even, middle line is repeated for last two figures.

2. **(10pt)** Write a program to print the multiplication table of the number entered by the user. The table should get displayed in the following form for n=29:

   29 * 1 = 29
   29 * 2 = 58
   29 * 3 = 87
   29 * 4 = 116
   …
   29 * 10 = 290

3. **(20pt)** Write a program to find the median and its index in a given array. The array should be entered by the user.
   The first entry from the user should be treated as the number of elements in the array, so that it should be a positive integer, and an error message should be displayed if it is not a positive integer before the program is terminated.
   If the first entry is a positive integer, following entries should be stored in a dynamically allocated memory as double type variables.
   When the user finishes to enter n numbers, if n is odd, program should display the median value, its index (e.g. 4 for an array of 1 2 4 3 5) and its address in the memory, than the program should terminate.
   If n is even, use the average of two related numbers (1 2 3 4 5 6, median= (3+4)/2 = 3.5), and display a message to indicate that index and memory address would be available only for arrays with an odd number of elements.

4. **(25pt)** Write a program that allocates memory for n number, stores the n numbers entered by the user, sorts them and displays the sorted array. The program should be called as "$./program_name n" from the terminal (for example as "./sort_array.x 15" to sort 15 numbers), request n number from the user, and make use of any sorting algorithm (such as bubble sort, insertion sort or quick sort) to sort them.

5. **(25pt)** Write a class to store a symmetric matrix only with its upper triangle values (including diagonal, and variable type double) with a default constructor and non-default parametrized constructor that takes an argument N and allocates memory for upper triangular part of N*N matrix **[5pt]** and a deconstructor that deallocates this memory, a copy constructor and overloaded assignment operator that handles corresponding operations **[5pt]**.

   This class should include a method, named "dot", that takes a memory address value (i.e. a pointer) as argument and performs a dot product with the matrix and N double values at the address token as argument. Note that N double values, contiguously stored at given address, will be treated as a vector (from a mathematical point of view) by the method function and a regular matrix-vector dot product will be performed.
   You should be aware of the fact that only certain values are stored in this class. So that some element-to-element multiplication operations in a regular matrix-vector dot product should be handled differently, by referring to different indices which stores same value

(for symmetric matrices) with regular matrix indices.
The class method function “dot” should store results contiguously, using the function “new” which is usually used for dynamic memory allocation, and return the starting address of that contiguous series of double variables as return value. **[15pt]**

Rules

- As with all programs in this course, your code should contain useful comments. In particular, your name, the date, and a brief description of what the program does should appear at the top of your source file.
- Each question should have its own folder.
- All files must be packed and compressed as a single file with following naming convention,
  hw-00-[STUDENT-ID].tar.gz
  The file must be uploaded to course Moodle site before its deadline:
  http://ninova.itu.edu.tr/
  You can use .zip compression or .tar or .tar.gz compression.

Hint

Following Linux commands can be used to create tar file

- tar -cvf hw-00-[STUDENT-ID].tar [DIRECTORY-TO-TAR]
- gzip hw-00-[STUDENT-ID].tar