Exercise 1. Answer Sheet

Student's Name: ______Tran Thi Thoa________ Student's ID: ______s1242006______

Problem 1. *(30 points)* For each function ***f(n)*** and time ***T*** in the following table, determine the largest size *n* of a problem that can be solved in time ***T****,* assuming that the algorithm to solve the problem takes ***f(n)*** milliseconds.

f(n)	***T*** = 1 second	***T*** = 1 minute	***T*** = 1 hour	***T*** = 1 day	***T*** = 1 month (30 days)
$\sqrt{n}$	10^12	36.10^14	1296.10^16	746496.10^16	6718464.10^18
n	10^6	6.10^7	36.10^8	864.10^8	2592.10^9
n^2	1000	7745	60000	293938	1609968
n^3	100	391	1532	4420	13736
2^n	19	25	31	36	41

Problem 2. *(30 points)* Consider sorting ***n*** numbers stored in array ***A*** by first finding the smallest element of ***A*** and exchanging it with the first element of the array, i.e. ***A[1]****.* Them find the second smallest element of ***A***, and exchange it with ***A[2]****.* Continue in this manner for the first *n-1* elements of *A*.

a) Write a pseudo-code for this algorithm which is known as "**Selection Sort**".

```
def selectionSort(A,n)
    //Input: an array A[1..n]
   //Output: a sorted array A[1..n]
    for i = 1 to n-1 do
        min = i;
        for j = i + 1 to n do
            if  A[j] < A[min] then
                  min = j;
           end if
        end for
        swap A[i], A[min]
    end for
```

b) What is the time complexity of the Selection Sort algorithm?

From the Selection Sort pseudo-code, we can see that the algorithm run through 2 nested loop. The sort method executes to find the smallest for the indexed from 1 to (n-1) and each time the function finding the smallest value is executed for an index, it does n-index comparisons

Therefore the time complexity of this algorithm is O(n^2).

Problem 3. *(40 points)* Using the pseudo-code for **Merge Sort** algorithm given at the lecture, write a program implementing the **Merge Sort** algorithm. Use any programming language you know. Upload your source code with instructions how to compile/run it. Give the input data and the program output in the space below.

This merge sort program was implemented in C

To run it, open the terminal and change the directory to the directory where you saved this file.

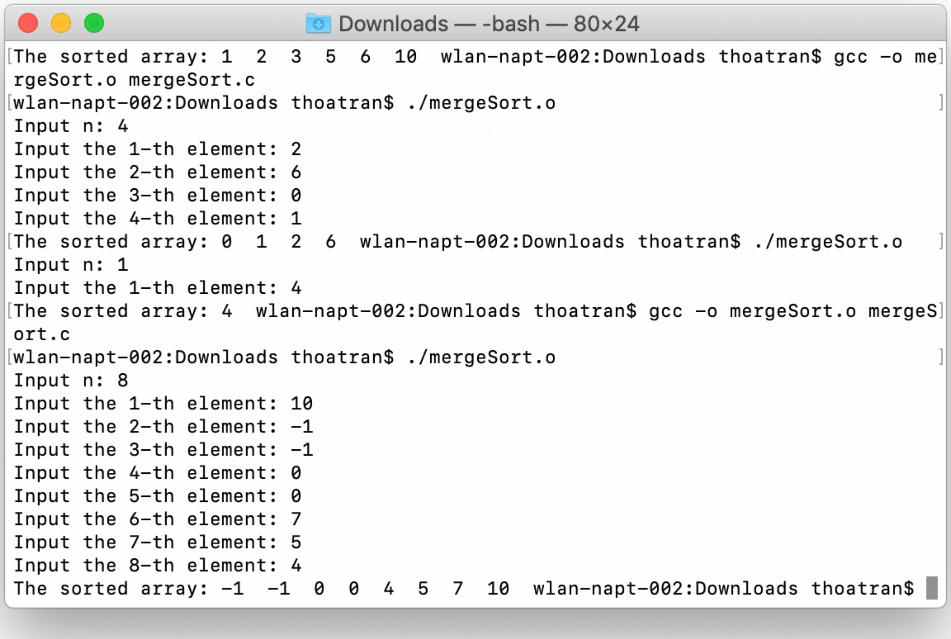
Run these command line:

```
gcc -o mergeSort.o mergeSort.c
./mergeSort.o
```

Input: first input is the number of elements in the array, then input each element of the array.

Output: The array after being sorted by mere sort algorithm.

For example:



```
[The sorted array: 1  2  3  5  6  10  wlan-napt-002:Downloads thoatran$ gcc -o me]
rgeSort.o mergeSort.c
[wlan-napt-002:Downloads thoatran$ ./mergeSort.o                                ]
Input n: 4
Input the 1-th element: 2
Input the 2-th element: 6
Input the 3-th element: 0
Input the 4-th element: 1
[The sorted array: 0  1  2  6  wlan-napt-002:Downloads thoatran$ ./mergeSort.o   ]
Input n: 1
Input the 1-th element: 4
[The sorted array: 4  wlan-napt-002:Downloads thoatran$ gcc -o mergeSort.o mergeS]
ort.c
[wlan-napt-002:Downloads thoatran$ ./mergeSort.o                                ]
Input n: 8
Input the 1-th element: 10
Input the 2-th element: -1
Input the 3-th element: -1
Input the 4-th element: 0
Input the 5-th element: 0
Input the 6-th element: 7
Input the 7-th element: 5
Input the 8-th element: 4
The sorted array: -1  -1  0  0  4  5  7  10  wlan-napt-002:Downloads thoatran$ 
```