

Exercise 2. Answer Sheet

Student's Name: ____Tran Thi Thoa____

Student's ID: ____s1242006____

Problem 1. (10 points) Consider a priority queue S implemented as a heap. Write a pseudo-code for the **Maximum(S)** operation on this priority queue.

```
def Maximum(S)
    // Input: heap S[1..n]
    // Return the element of S with the largest key
    return S[1];
```

Problem 2. (20 points) Consider top-down heap construction approach.

a). Write a pseudo-code for a **HeapTopDown(A)** algorithm using **Max-Heap-Insert (A, key)** operation

```
def Max-Heap-Insert(A, key)
    //Input: heap A[1..n] and new key.
    //Output: heap A[1..n+1]
    A.heap_size = A.heap_size + 1
    A[A.heap_size] = MAX_NEGATIVE
    Heap-Increase-Key(A, A.heap_size, key)

def HeapTopDown(A)
    //constructs a heap from the elements of a given array by the top-down algorithm
    //Input: An array A[1..n]
    //Output: A heap A[1..n]
    B.length = A.length
    for i = 1 to B.length
        B[i] = A[i]
    end for
    A.heap_size = 0
    for i = 1 to B.length
        A = Max-Heap-Insert(A,B[i])
    end for
```

b) What is the time complexity of **HeapTopDown(A)** algorithm? Why?

From the algorithm of the HeapTopDown(A) , we can see that the algorithm has 2 part:

+ Array B construction which has $O(n)$ time complexity.

+ The algorithm repeats n times the Max-Heap-Insert(A , key) algorithm, which has the time complexity is $O(\log n)$.

Total: $O(n) + n O(\log n) = O(n \log n)$

Therefore, the time complexity of the HeapTopDown(A) algorithm is $O(n \log n)$

Problem 3. (20 points) Illustrate the operation **Heap-Extract-Max** on a heap $A=[15,13,9,5,12,8,7,4,0,6]$

The pseudo-code for the Heap-Extract-Max() algorithm is:

```
Def Heap-Extract-Max(A)
    //Input: heapA[1..n]
    //Removes and returns the root element
    max = A[1]
    A[1] = A[A.heap_size]
    A.heap_size = A.heap_size - 1
    MaxHeapify(A,1)
    return max
```

The Heap-Extract-Max operations on the heap A :

```
A = [15,13,9,5,12,8,7,4,0,6]      //define max = A[1]
max = 15
```

```
A = [6,13,9,5,12,8,7,4,0]        //let A[1] = A[heap_size]
max = 15                          //and decrease A.heap_size by 1
```

```
A = [13,6,9,5,12,8,7,4,0]        //call MaxHeapify(A,1)
max = 15
```

```
A = [13,12,9,5,6,8,7,4,0]        //call MaxHeapify(A,2)
max = 15
```

```
A = [13,12,9,5,6,8,7,4,0]        //call MaxHeapify(A,5) and return max = 15
max = 15
```

The operations finish with the result removing the root element from the heap and return its value(in the example, the root element has the value 15)

Problem 4. (50 points) Write a program implementing **HeapBottomUp (A)** algorithm. Upload your source code. Show your input array and the output heap in the space below.

Implement Heap Bottom Up algorithm in C programming language

To run this program, open terminal and change the directory to the folder where you saved this file

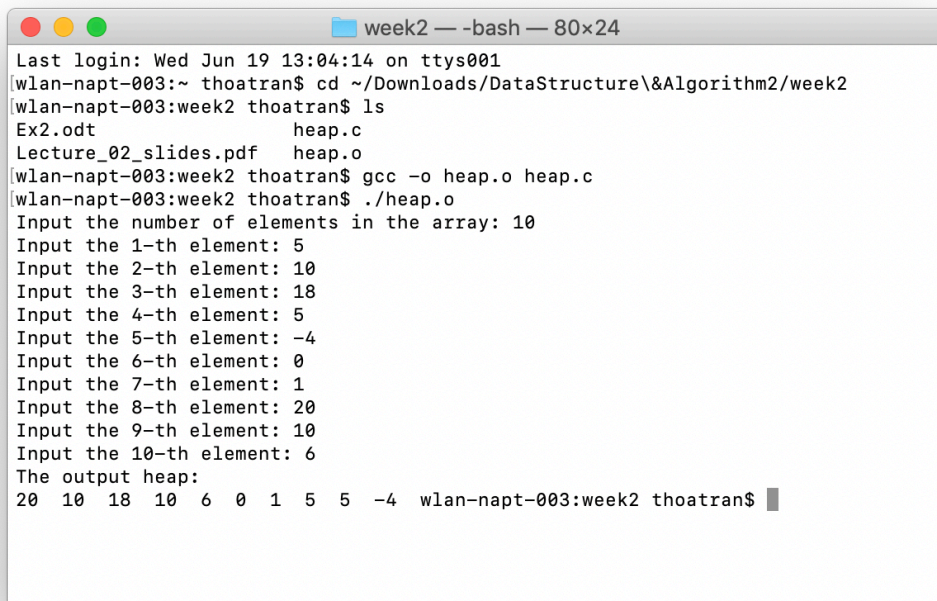
Run the following command line

```
gcc -o heap.o heap.c
./heap.o
```

Input: First input is the number of element in the array, then input each element in the array respectively

Output: The heap using HeapBottomUp algorithm

For example:

A terminal window titled "week2 — -bash — 80x24" showing the execution of a C program. The user navigates to the directory ~/Downloads/DataStructure&Algorithm2/week2 and lists files, including heap.c and heap.o. They compile heap.c into heap.o using gcc. Then, they run ./heap.o, which prompts for the number of elements (10) and then for each element. The final output is the heap array: 20 10 18 10 6 0 1 5 5 -4.

```
week2 — -bash — 80x24
Last login: Wed Jun 19 13:04:14 on ttys001
[wlan-napt-003:~ thoatran$ cd ~/Downloads/DataStructure&Algorithm2/week2
[wlan-napt-003:week2 thoatran$ ls
Ex2.odt          heap.c
Lecture_02_slides.pdf  heap.o
[wlan-napt-003:week2 thoatran$ gcc -o heap.o heap.c
[wlan-napt-003:week2 thoatran$ ./heap.o
Input the number of elements in the array: 10
Input the 1-th element: 5
Input the 2-th element: 10
Input the 3-th element: 18
Input the 4-th element: 5
Input the 5-th element: -4
Input the 6-th element: 0
Input the 7-th element: 1
Input the 8-th element: 20
Input the 9-th element: 10
Input the 10-th element: 6
The output heap:
20 10 18 10 6 0 1 5 5 -4 wlan-napt-003:week2 thoatran$
```