Data Structures and Algorithms

Lab Report

Lab12

Group Members Name & Reg #:	**Muhammad Haris Irfan** **(FA18-BCE-090)**
Class	Data Structures and Algorithms CSC211 (**BCE-3B**)
Instructor's Name	Dilshad Sabir

In Lab Tasks

Task:1

Complete the function 'void insert_node (int x, struct heap_struct * H)' in the skeleton code provided.

Solution:

The code is shown below,

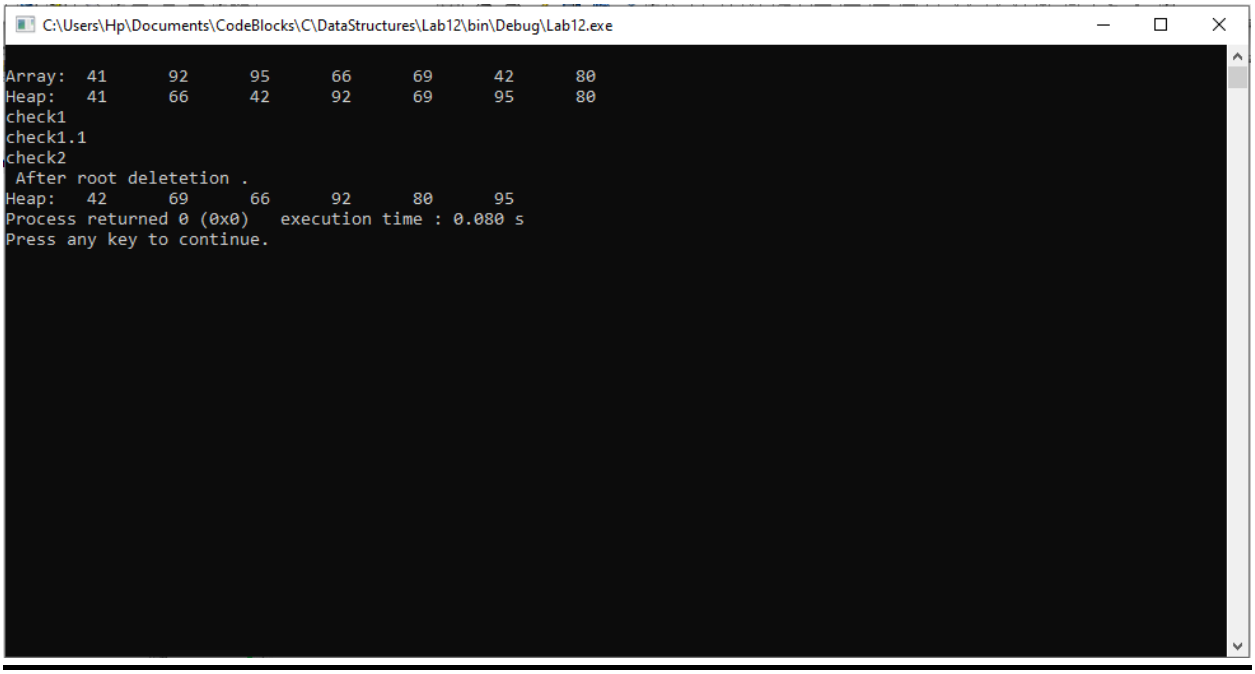
```
void insert_node(int x, struct heap_struct * H)
{
    unsigned int i;
    i = 0;
    int current = H->crnt_heap_size;
    int max = H->max_heap_size;

    if(is_full(H))
    {
        printf("\nError! Heap is full !!\n");
        return;
    }
    else
    {
        while( i <= current)
        {
            i++;
        }
            int z = i;
            H->elements[i] = x;
            H->crnt_heap_size = current +1;

        if (i > 1)
        {
            while ((H->elements[i]) < H->elements[i/2] )
            {
                int temp = H->elements[i/2];
                H->elements[i/2] = H->elements[i];
                H->elements[i] = temp;
                i = i/2;

                if (i < 1)
                {                       break;
                }}}}}
```

The Result of the following code is attached below:


C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab12\bin\Debug\Lab12.exe
Array: 41   92   95   66   69   42   80
Heap: 41   66   42   92   69   95   80
check1
check1.1
check2
 After root deletetion .
Heap: 42   69   66   92   80   95
Process returned 0 (0x0)   execution time : 0.080 s
Press any key to continue.

Task:2

Complete the function 'void delete_root (struct heap_struct * H)' in the skeleton code provided.

Solution:

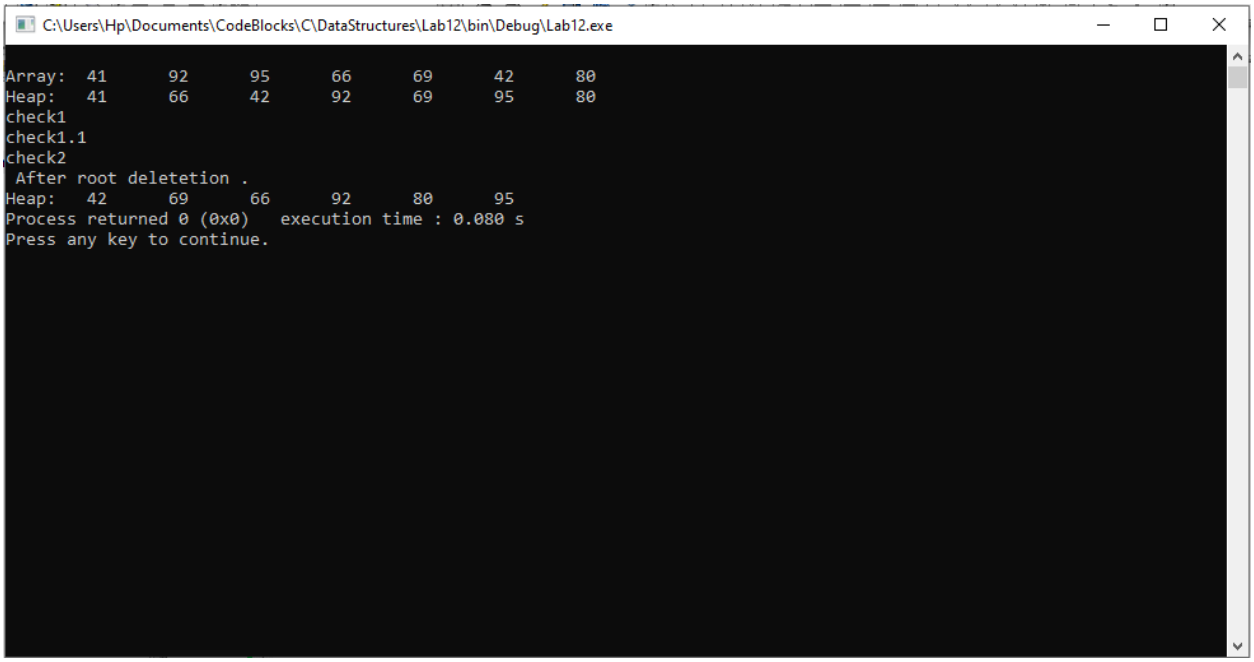
The code is shown below,

```
void delete_root(struct heap_struct * H)
{    /** Complete this function **/
  int size,sizel;
    size = H->max_heap_size;
    sizel=H->crnt_heap_size;
    int temp;
    H->elements[1] = H->elements[size];
    H->elements[size] = -1;
    H->crnt_heap_size = sizel-1;
    int i,z;
    i = 1;
    z = 1;
        while(H->elements[i] > H->elements[2*i] || H->elements[i] > H->elements[2*i+1] && H->elements[i] != -1)
        {
            if(H->elements[i] > H->elements[2*i])
            {
                temp = H->elements[i];
                H->elements[i] = H->elements[2*i];
                H->elements[2*i] = temp;
                i = 2*i;
                printf("\ncheck1");
            }
            else
            {
                temp = H->elements[i];
                H->elements[i] = H->elements[2*i+1];
                H->elements[2*i+1] = temp;
                i = 2*i+1;
                printf("\ncheck1.1");
            }
            if(i>size)
            {
                break;
            }
        }

        while(H->elements[z] > H->elements[2*z + 1] && H->elements[z] != -1)
        {
            temp = H->elements[z];
            H->elements[z] = H->elements[2*z+1];
            H->elements[2*z+1] = temp;
            i= 2*z+1;
            printf("\ncheck2");
            if(i>size)
            {
                break;
            }
        }


}
```

The Result of the following code is attached below:


C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab12\bin\Debug\Lab12.exe
Array: 41 92 95 66 69 42 80
Heap: 41 66 42 92 69 95 80
check1
check1.1
check2
After root deletetion .
Heap: 42 69 66 92 80 95
Process returned 0 (0x0) execution time : 0.080 s
Press any key to continue.

Post Lab Task.

Task 2:

Study Binomial Heaps and Fibonacci Heaps

Solution

Binomial Heap

A Binomial heap is a data structure that acts as a priority queue but also allows pairs of heaps to be merged together, they are implemented as a set of binomial trees.

Fibonacci Heap

A Fibonacci heap is a collection of trees satisfying the minimum-heap property, that is, the key of a child is always greater than or equal to the key of the parent. This implies that the minimum key is always at the root of one of the trees. Compared with binomial heaps, the structure of a Fibonacci heap is more flexible.

Conclusion:

In this lab, we completed the insertion and deletion function of a Binary Heap, furthermore we also studied Binomial and Fibonacci Heaps and the difference between them.

THE END