Shaheed Zulfikar Ali Bhutto Institute of Science & Technology

COMPUTER SCIENCE DEPARTMENT

Total Marks: 7.5

Obtained Marks: ________

DATA STRUCTURE AND ALGORITHM

Lab Report # 11

Submitted To: **Mam Tehreem**

Submitted By: _____**Hammad Qureshi** ___________________.

Reg. Numbers: **2112114**

Question no 1:

Heap (Max-Heap, Min-Heap, Insertion and Deletion)

Code:

Max Heap

```
#include <iostream>
using namespace std;
void max_heap(int *a, int m, int n) {
  int j, t;
  t = a[m];
  j = 2 * m;
  while (j <= n) {
    if (j < n && a[j+1] > a[j])
      j = j + 1;
    if (t > a[j])
      break;
    else if (t <= a[j]) {
      a[j / 2] = a[j];
      j = 2 * j;
    }
  }
  a[j/2] = t;
  return;
}
void build_maxheap(int *a,int n) {
  int k;
  for(k = n/2; k >= 1; k--) {
    max_heap(a,k,n);
  }
}
int main() {
  int n, i;
  cout<<"enter no of elements of array\n";
  cin>>n;
  int a[30];
  for (i = 1; i <= n; i++) {
    cout<<"enter elements"<<" "<<(i)<<endl;
    cin>>a[i];
  }
  build_maxheap(a,n);
  cout<<"Max Heap\n";
  for (i = 1; i <= n; i++) {
    cout<<a[i]<<endl;
  }
}
```

Min heap

```
#include <iostream>
#include <conio.h>
using namespace std;
void min_heap(int *a, int m, int n){
  int j, t;
  t= a[m];
  j = 2 * m;
  while (j <= n) {
    if (j < n && a[j+1] < a[j])
```

```
      j = j + 1;
    if (t < a[j])
      break;
    else if (t >= a[j]) {
      a[j/2] = a[j];
      j = 2 * j;
    }
  }
  a[j/2] = t;
  return;
}
void build_minheap(int *a, int n) {
  int k;
  for(k = n/2; k >= 1; k--) {
    min_heap(a,k,n);
  }
}
int main() {
  int n, i;
  cout<<"enter no of elements of array\n";
  cin>>n;
  int a[30];
  for (i = 1; i <= n; i++) {
    cout<<"enter element"<<" "<<(i)<<endl;
    cin>>a[i];
  }
  build_minheap(a, n);
  cout<<"Min Heap\n";
  for (i = 1; i <= n; i++) {
    cout<<a[i]<<endl;
  }
  getch();
}
```

Insertion

```
#include <iostream>
using namespace std;

#define MAX 1000 // Max size of Heap

// Function to heapify ith node in a Heap
// of size n following a Bottom-up approach
void heapify(int arr[], int n, int i)
{
    // Find parent
    int parent = (i - 1) / 2;

    if (arr[parent] > 0) {
        // For Max-Heap
        // If current node is greater than its parent
        // Swap both of them and call heapify again
        // for the parent
        if (arr[i] > arr[parent]) {
            swap(arr[i], arr[parent]);

            // Recursively heapify the parent node
            heapify(arr, n, parent);
        }
    }
```

```
}

// Function to insert a new node to the Heap
void insertNode(int arr[], int& n, int Key)
{
   // Increase the size of Heap by 1
   n = n + 1;

   // Insert the element at end of Heap
   arr[n - 1] = Key;

   // Heapify the new node following a
   // Bottom-up approach
   heapify(arr, n, n - 1);
}

// A utility function to print array of size n
void printArray(int arr[], int n)
{
   for (int i = 0; i < n; ++i)
      cout << arr[i] << " ";

   cout << "\n";
}

// Driver Code
int main()
{
   // Array representation of Max-Heap
```

```
    // 10
    //    / \
    // 5    3
    //  / \
    // 2   4
    int arr[MAX] = { 10, 5, 3, 2, 4 };

    int n = 5;

    int key = 15;

    insertNode(arr, n, key);

    printArray(arr, n);
    // Final Heap will be:
    // 15
    //    /   \
    // 5     10
    //  / \   /
    // 2   4 3
    return 0;
}
```

Deletion

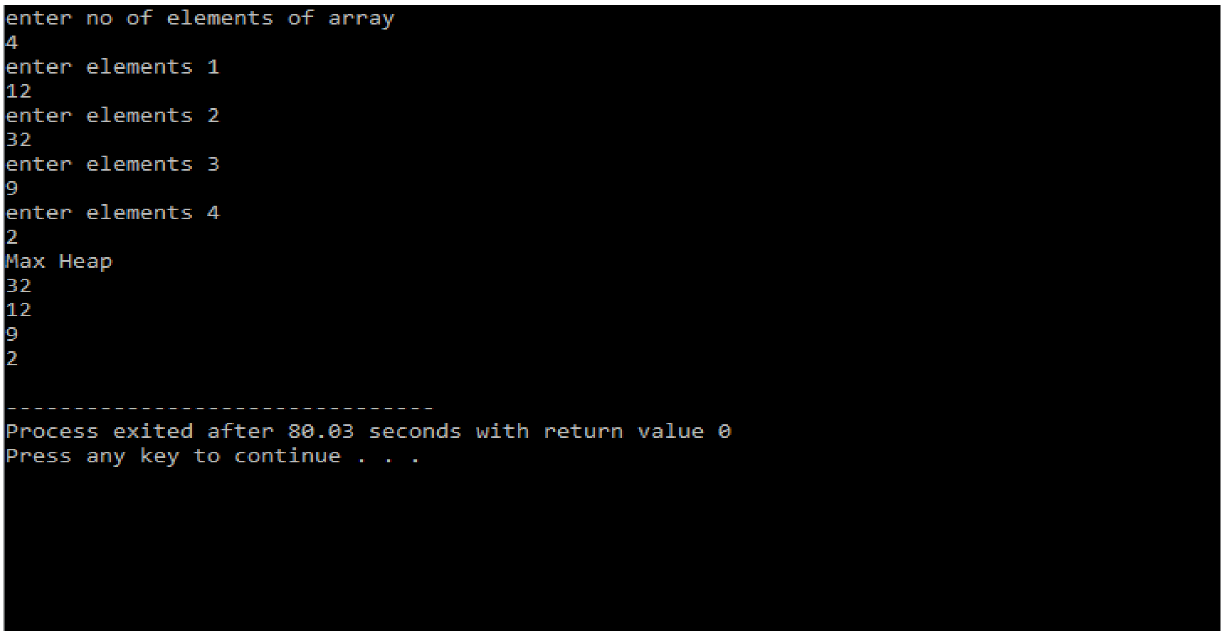
```
#include <iostream>

using namespace std;

// To heapify a subtree rooted with node i which is
// an index of arr[] and n is the size of heap
```

```
void heapify(int arr[], int n, int i)
{
   int largest = i; // Initialize largest as root
   int l = 2 * i + 1; // left = 2*i + 1
   int r = 2 * i + 2; // right = 2*i + 2

   // If left child is larger than root
   if (l < n && arr[l] > arr[largest])
      largest = l;

   // If right child is larger than largest so far
   if (r < n && arr[r] > arr[largest])
      largest = r;

   // If largest is not root
   if (largest != i) {
      swap(arr[i], arr[largest]);

      // Recursively heapify the affected sub-tree
      heapify(arr, n, largest);
   }
}

// Function to delete the root from Heap
void deleteRoot(int arr[], int& n)
{
   // Get the last element
   int lastElement = arr[n - 1];
```

```
    // Replace root with last element
    arr[0] = lastElement;

    // Decrease size of heap by 1
    n = n - 1;

    // heapify the root node
    heapify(arr, n, 0);
}

/* A utility function to print array of size n */
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; ++i)
        cout << arr[i] << " ";
    cout << "\n";
}

// Driver Code
int main()
{
    // Array representation of Max-Heap
    //     10
    //    /  \
    //   5    3
    //  / \
    // 2   4
    int arr[] = { 10, 5, 3, 2, 4 };
    int n = sizeof(arr) / sizeof(arr[0]);

    deleteRoot(arr, n);

    printArray(arr, n);

    return 0;
}
```

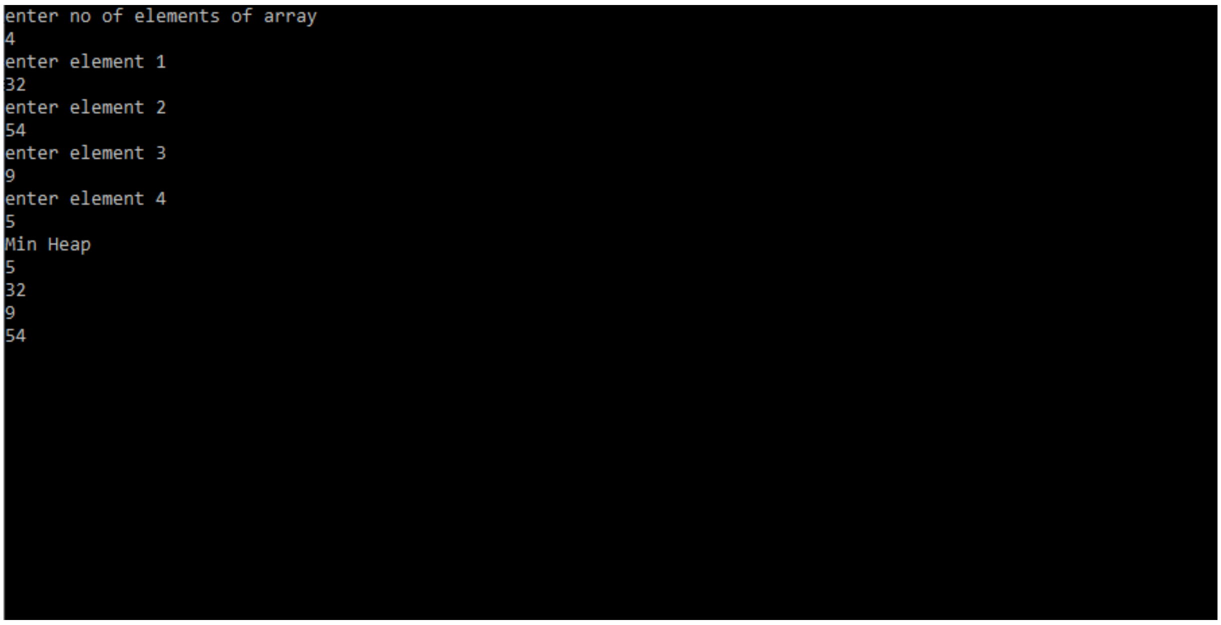
CONSOLE SCREEN:

Max heap



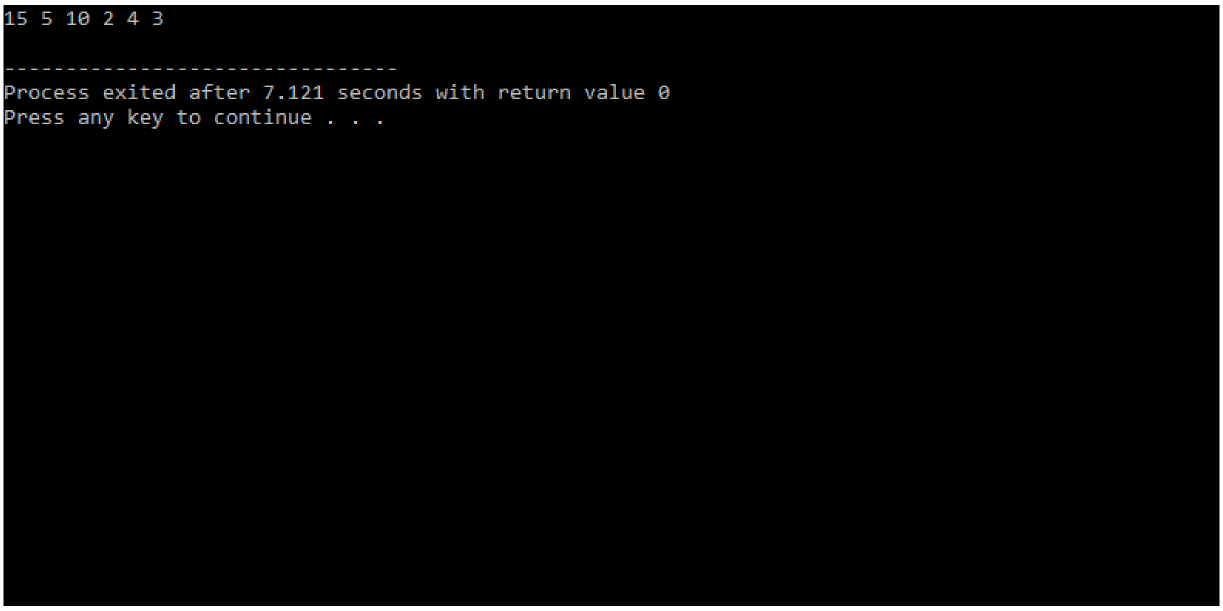
```
enter no of elements of array
4
enter elements 1
12
enter elements 2
32
enter elements 3
9
enter elements 4
2
Max Heap
32
12
9
2

--------------------------------
Process exited after 80.03 seconds with return value 0
Press any key to continue . . .
```

Min heap



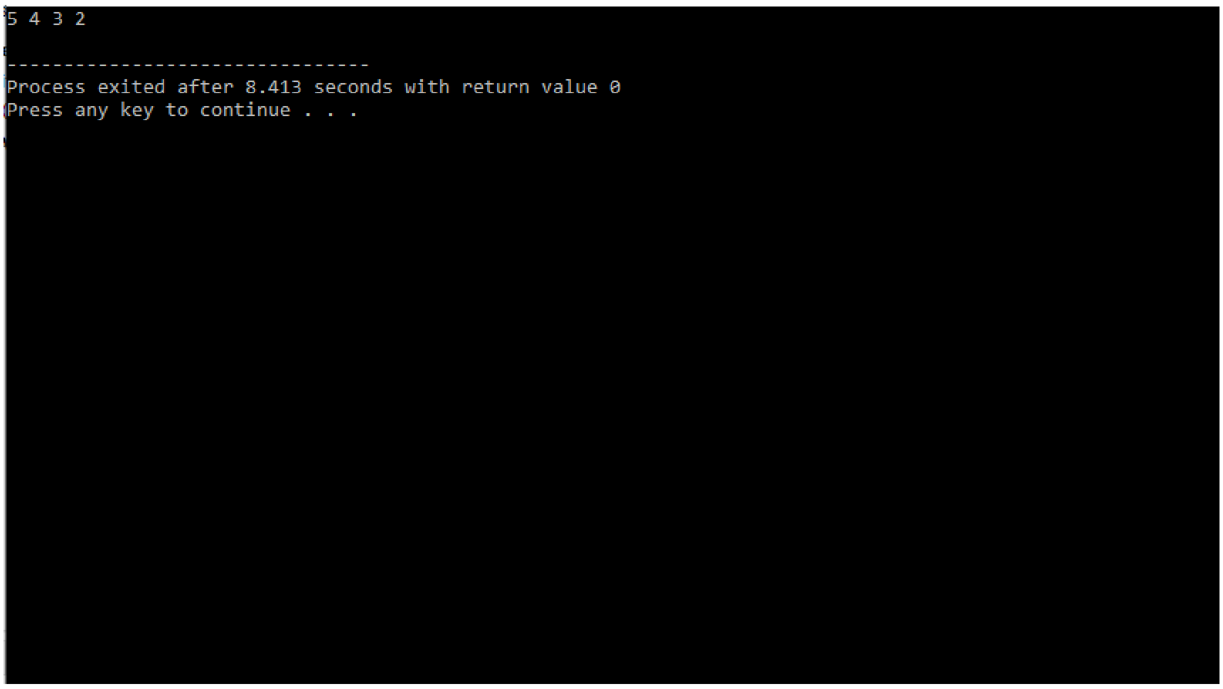
```
enter no of elements of array
4
enter element 1
32
enter element 2
54
enter element 3
9
enter element 4
5
Min Heap
5
32
9
54
```

Insertion



```
15 5 10 2 4 3

--------------------------------
Process exited after 7.121 seconds with return value 0
Press any key to continue . . .
```

Deletion



```
5 4 3 2

--------------------------------
Process exited after 8.413 seconds with return value 0
Press any key to continue . . .
```