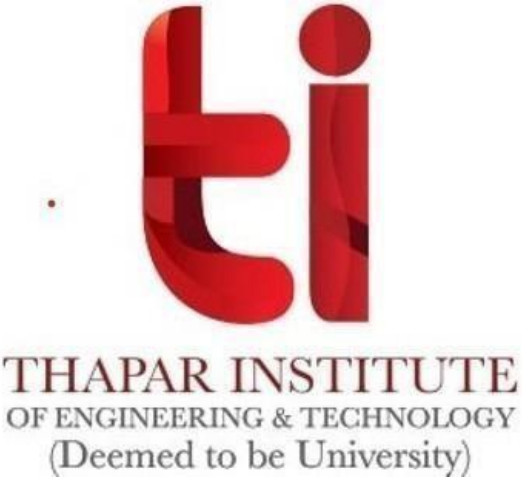
A Practical Activity Report For
Data Structures and Algorithms (UCS406)

Submitted By: **Vivek Arora**

101715178
(ENC 8)

Submitted To:

Dr. Sanjay Sharma



ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING &TECHNOLOGY, (DEEMED TO BEUNIVERSITY), PATIALA, PUNJAB

ASSIGNMENT 7

QUESTION 1(Various Functions of Circular Queue)

Create circular queue using arrays .Perform following functions:

- **i)** Insert or enqueue
- **ii)** Remove or dequeue
- **iii)** isFull
- **iv)** isempty

```
#include<stdio.h>
int queue[100];
int front;
int rear;
int n;
void enqueue(int item)
{
   rear=(rear+1)%n;
   if(front==rear)
   {
        printf("Queue is full");
        if(rear==0)
        {
                rear=n-1;
                }
                else
                rear=rear-1;
                return;
        }
        else
        {
                queue[rear]=item;
           return;
        }

}
int dequeue()
{
        if(front==rear)
        {
                printf("Queue is empty");
                return -1;
```

```
        }
        else
        {
                front=(front+1)%n;//front=(front+1)%n
                item=queue[front];
                return item;
        }
}
int main()
{   printf("Enter size of queue");
    scanf("%d",&n);
    int item;
    printf("Enter number to be inserted:");
        scanf("%d",&item);
        enqueue(item);
    int removedno=dequeue();
        printf("removed  no was",removedno);
}
```

QUESTION 2 Create a heap data structure using arrays . Implement insert function and delete function for heap.

```
#include <iostream>
#include <cstdlib>
#include <vector>
#include <iterator>
using namespace std;
class BHeap {
   private:
   vector <int> heap;
   int l(int parent);
   int r(int parent);
   int par(int child);
   void heapifyup(int index);
   void heapifydown(int index);
   public:
      BHeap() {}
      void Insert(int element);
      void DeleteMin();
      int ExtractMin();
```

```
    void showHeap();
    int Size();
};
int main() {
  BHeap h;
  while (1) {
    cout<<"1.Insert Element"<<endl;
    cout<<"2.Delete Minimum Element"<<endl;
    cout<<"3.Extract Minimum Element"<<endl;
    cout<<"4.Show Heap"<<endl;
    cout<<"5.Exit"<<endl;
    int c, e;
    cout<<"Enter your choice: ";
    cin>>c;
    switch(c) {
      case 1:
        cout<<"Enter the element to be inserted: ";
        cin>>e;
        h.Insert(e);
      break;
      case 2:
        h.DeleteMin();
      break;
      case 3:
        if (h.ExtractMin() == -1) {
          cout<<"Heap is Empty"<<endl;
        }
        else
        cout<<"Minimum Element: "<<h.ExtractMin()<<endl;
      break;
      case 4:
        cout<<"Displaying elements of Hwap: ";
        h.showHeap();
      break;
      case 5:
        exit(1);
        default:
        cout<<"Enter Correct Choice"<<endl;
    }
  }
  return 0;
}
int BHeap::Size() {
```

```
  return heap.size();
}
void BHeap::Insert(int ele) {
  heap.push_back(ele);
  heapifyup(heap.size() -1);
}
void BHeap::DeleteMin() {
  if (heap.size() == 0) {
    cout<<"Heap is Empty"<<endl;
    return;
  }
  heap[0] = heap.at(heap.size() - 1);
  heap.pop_back();
  heapifydown(0);
  cout<<"Element Deleted"<<endl;
}

int BHeap::ExtractMin() {
  if (heap.size() == 0) {
    return -1;
  }
  else
  return heap.front();
}

void BHeap::showHeap() {
  vector <int>::iterator pos = heap.begin();
  cout<<"Heap --> ";
  while (pos != heap.end()) {
    cout<<*pos<<" ";
    pos++;
  }
  cout<<endl;
}

int BHeap::l(int parent) {
  int l = 2 * parent + 1;
  if (l < heap.size())
    return l;
  else
    return -1;
}
int BHeap::r(int parent) {
```

```
  int r = 2 * parent + 2;
  if (r < heap.size())
    return r;
  else
    return -1;
}
int BHeap::par(int child) {
  int p = (child - 1)/2;
  if (child == 0)
    return -1;
  else
    return p;
}

void BHeap::heapifyup(int in) {
  if (in >= 0 && par(in) >= 0 && heap[par(in)] > heap[in]) {
    int temp = heap[in];
    heap[in] = heap[par(in)];
    heap[par(in)] = temp;
    heapifyup(par(in));
  }
}

void BHeap::heapifydown(int in) {
  int child = l(in);
  int child1 = r(in);
  if (child >= 0 && child1 >= 0 && heap[child] > heap[child1]) {
    child = child1;
  }
  if (child > 0 && heap[in] > heap[child]) {
    int t = heap[in];
    heap[in] = heap[child];
    heap[child] = t;
    heapifydown(child);
  }
}
```

QUESTION 3 – Given an array . Heapify the array elements to build a MAX-HEAP

```
#include <iostream>
#include <conio.h>
using namespace std;
void max_heapify(int *a, int i, int n)
{
   int j, temp;
   temp = a[i];
   j = 2 * i;
   while (j <= n)
   {
      if (j < n && a[j+1] > a[j])
         j = j + 1;
      if (temp > a[j])
         break;
      else if (temp <= a[j])
      {
         a[j / 2] = a[j];
         j = 2 * j;
      }
   }
   a[j/2] = temp;
   return;
}
void build_maxheap(int *a,int n)
{
   int i;
   for(i = n/2; i >= 1; i--)
   {
      max_heapify(a,i,n);
   }
}
int main()
{
   int n, i, x;
   cout<<"enter no of elements of array\n";
   cin>>n;
   int a[20];
   for (i = 1; i <= n; i++)
   {
```

```
        cout<<"enter element"<<(i)<<endl;
        cin>>a[i];
    }
    build_maxheap(a,n);
    cout<<"Max Heap\n";
    for (i = 1; i <= n; i++)
    {
        cout<<a[i]<<endl;
    }

}
```

QUESTION 4 Given an array . Heapify the array elements to build a MIN-HEAP

```
#include <iostream>
#include <conio.h>
using namespace std;
void min_heapify(int *a,int i,int n)
{
    int j, temp;
    temp = a[i];
    j = 2 * i;
    while (j <= n)
    {
        if (j < n && a[j+1] < a[j])
            j = j + 1;
        if (temp < a[j])
            break;
        else if (temp >= a[j])
        {
            a[j/2] = a[j];
            j = 2 * j;
        }
    }
    a[j/2] = temp;
    return;
}
void build_minheap(int *a, int n)
{
    int i;
    for(i = n/2; i >= 1; i--)
```

```
    {
        min_heapify(a,i,n);
    }
}
int main()
{
    int n, i, x;
    cout<<"enter no of elements of array\n";
    cin>>n;
    int a[20];
    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;
    }

}

}
```

QUESTION 5 Write a program to HEAP–SORT an array of integer values.

```
#include<iostream>
using namespace std;
void heapify(int arr[], int n, int i) {
  int temp;
  int largest = i;
  int l = 2 * i + 1;
  int r = 2 * i + 2;
  if (l < n && arr[l] > arr[largest])
    largest = l;
  if (r < n && arr[r] > arr[largest])
    largest = r;
  if (largest != i) {
```

```
    temp = arr[i];
    arr[i] = arr[largest];
    arr[largest] = temp;
    heapify(arr, n, largest);
  }
}
void heapSort(int arr[], int n) {
  int temp;
  for (int i = n / 2 - 1; i >= 0; i--)
    heapify(arr, n, i);
  for (int i = n - 1; i >= 0; i--) {
    temp = arr[0];
    arr[0] = arr[i];
    arr[i] = temp;
    heapify(arr, i, 0);
  }
}
int main() {
  int arr[] = { 20, 7, 1, 54, 10, 15, 90, 23, 77, 25};
  int n = 10;
  int i;
  cout<<"Given array is: "<<endl;
  for (i = 0; i *lt; n; i++)
    cout<<arr[i]<<" ";
  cout<<endl;
  heapSort(arr, n);
  printf("\nSorted array is: \n");
  for (i = 0; i < n; ++i)
    cout<<arr[i]<<" ";
    return 0;
}
```