```
/*WAP to 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);
    }
}
```