Q. Implement Heap Sort to sort given set of values using max or min heap. Analyse the implemented algorithm for space and time complexity.

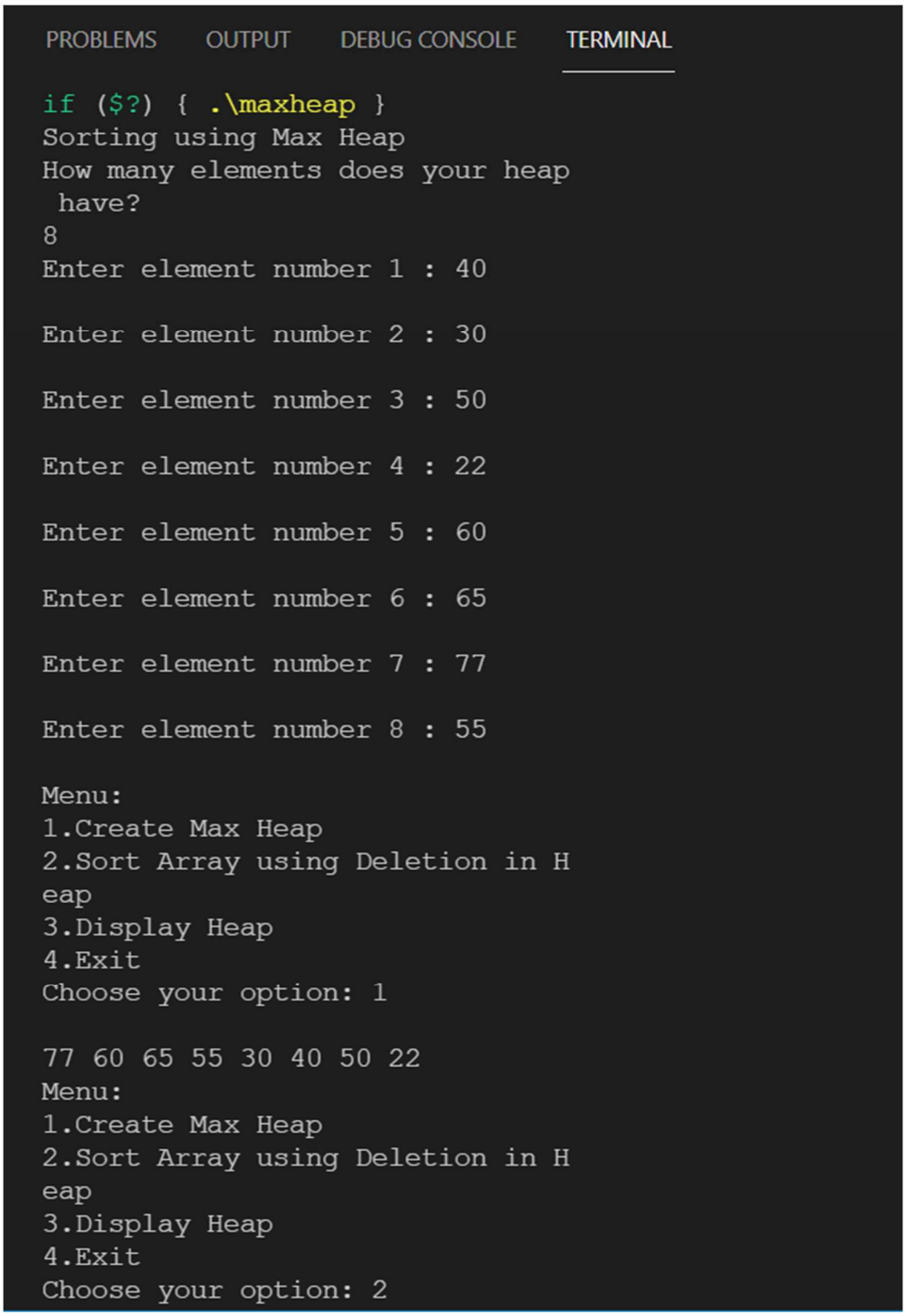
Code

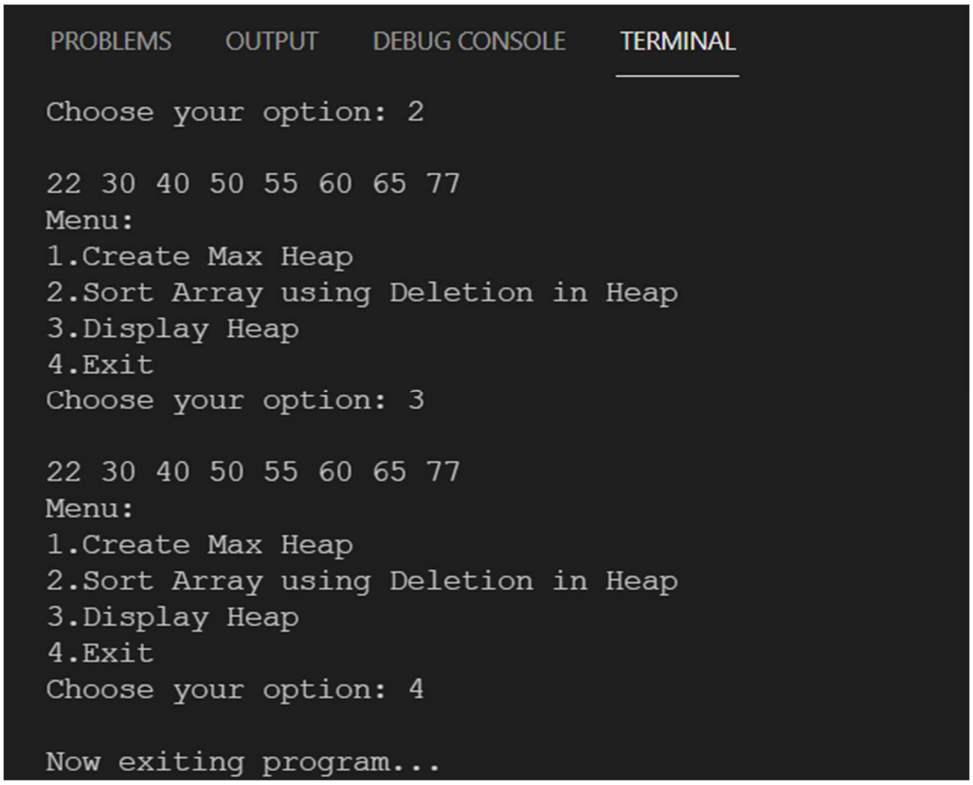
```
#include <iostream>
#include <cmath>
using namespace std;

class Max_Heap{
    private:
    int *heap;
    int n;
    int input;

    public:
    Max_Heap();
    void Heapify (int i, int n); //i is position of element and n is
size of array
    void Insert_Element();
    bool isFull();
    void createMaxHeap();
    void deleteHeap();
    void Display();
};

Max_Heap ::Max_Heap(){
    cout <<"How many elements does your heap have? "<<endl;
    cin >> n;
    heap = new int[n+1];
    heap[0] = 0;  //so that array starts from 1
    for (int j = 1;j<n+1;j++){
        cout << "Enter element number "<<j<<" : ";
        cin >> heap[j];
        cout<<"\n";
    }
}
void Max_Heap :: Heapify(int c, int n){

    int largest = c;
    int left_child = (2*c);
    int right_child = (2*c)+1;

    //comparing with right and left child
    while (left_child <=n && heap[left_child]> heap[largest]){
        largest = left_child;
    }
    while (right_child <=n && heap[right_child]> heap[largest]){
        largest = right_child;
    }
    //if one of the children is greater then swapping and calling
heapify again on new subtree
```

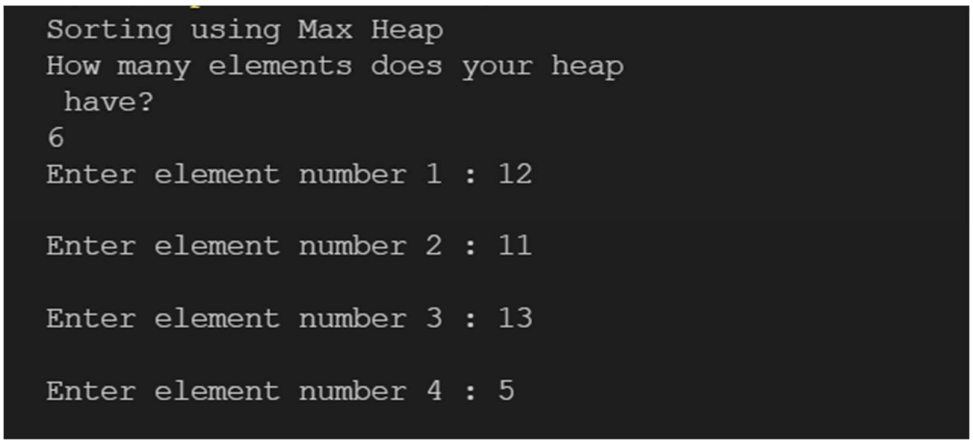
```
    if (largest != c){
        swap(heap[largest],heap[c]);
        Heapify(largest,n);
    }
}

void Max_Heap ::Display(){
    for (int j=1;j<=n;j++){
        cout<<heap[j]<<" ";
    }
    cout<<"\n";
}

void Max_Heap :: createMaxHeap(){
    for (int i = n/2;i>=1;i--){
        Heapify(i,n);
    }
}

void Max_Heap ::deleteHeap(){
    for (int i = n;i>=1;){
        swap(heap[1],heap[i]);
        i--;  //Reducing size of array before calling heapify
        Heapify(1,i);
    }
}

int main(){
    cout << "Sorting using Max Heap"<<endl;
    Max_Heap maxheap;
    while (1){
        cout <<"Menu:\n1.Create Max Heap\n2.Sort Array using Deletion
in Heap\n3.Display Heap\n4.Exit"<<endl;
        int ch;
        cout<<"Choose your option: ";
        cin >> ch;
        cout <<"\n";
        switch (ch){
            case 1:
            maxheap.createMaxHeap();
            maxheap.Display();
            break;
            case 2:
            maxheap.deleteHeap();
            maxheap.Display();
            break;
            case 3:
            maxheap.Display();
```

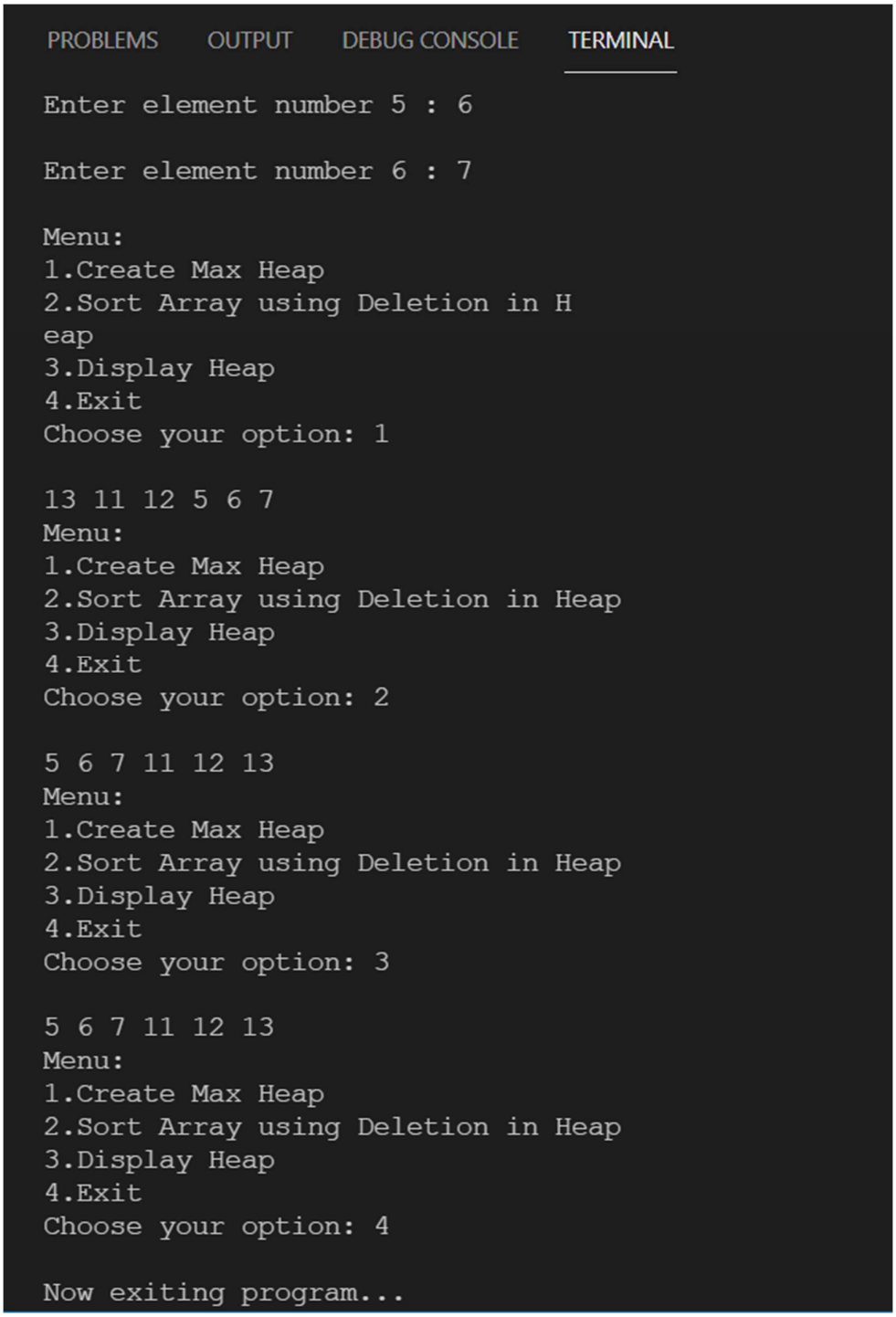
```
            break;
            case 4:
            cout<<"Now exiting program..."<<endl;
            exit(0);
        }
    }

}
```

Sample Output




PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Choose your option: 2
22 30 40 50 55 60 65 77
Menu:
1.Create Max Heap
2.Sort Array using Deletion in Heap
3.Display Heap
4.Exit
Choose your option: 3
22 30 40 50 55 60 65 77
Menu:
1.Create Max Heap
2.Sort Array using Deletion in Heap
3.Display Heap
4.Exit
Choose your option: 4
Now exiting program...


Sorting using Max Heap
How many elements does your heap
have?
6
Enter element number 1 : 12
Enter element number 2 : 11
Enter element number 3 : 13
Enter element number 4 : 5


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Enter element number 5 : 6
Enter element number 6 : 7
Menu:
1.Create Max Heap
2.Sort Array using Deletion in H
eap
3.Display Heap
4.Exit
Choose your option: 1
13 11 12 5 6 7
Menu:
1.Create Max Heap
2.Sort Array using Deletion in Heap
3.Display Heap
4.Exit
Choose your option: 2
5 6 7 11 12 13
Menu:
1.Create Max Heap
2.Sort Array using Deletion in Heap
3.Display Heap
4.Exit
Choose your option: 3
5 6 7 11 12 13
Menu:
1.Create Max Heap
2.Sort Array using Deletion in Heap
3.Display Heap
4.Exit
Choose your option: 4
Now exiting program...