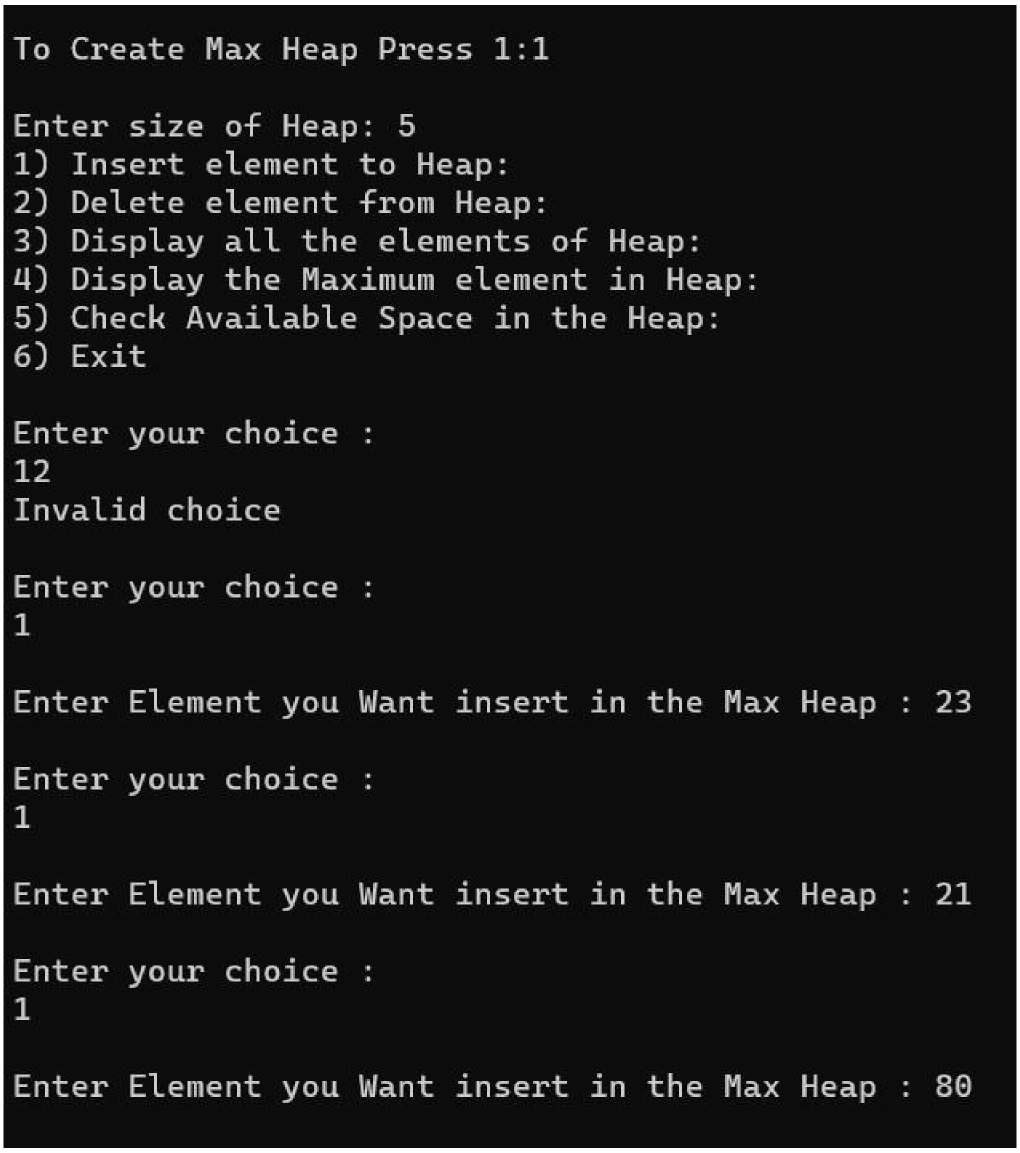
MaxHeap.cpp

```
#include <iostream>
#include<stdlib.h>
#include<conio.h>
using namespace std;
class BinaryMaxHeap
{
        public:
         int *data;
         int heapsize;
         int arraysize;
         BinaryMaxHeap(int size)
         {
                  data=new int[size];
                 heapsize=0;
                 arraysize=size;
         }
         int getLeftChildIndex(int node);
         int getRightChildIndex(int node);
         int getParentChildIndex(int node);
         void display();
         void insert(int val);
         void reheapUp(int node);
         void remove();
         void checkSpace();
         void reheapDown(int node);
         int getMax();
};
int BinaryMaxHeap::getLeftChildIndex(int node)
{
        return((2*node)+1);
}
int BinaryMaxHeap::getRightChildIndex(int node)
{
         return((2*node)+2);
}
int BinaryMaxHeap::getParentChildIndex(int node)
{
        return((node-1)/2);
}
void BinaryMaxHeap::display()
{
        for(int i=0;i<heapsize;i++)
         {
                cout<<data[i]<<" ";
         }
}
void BinaryMaxHeap::insert(int val)
{
         if(heapsize==arraysize)
         {
                cout<<"\nSorry!!! We Can't Put "<< val <<" Because Heap is Full.";
         }
         else
         {
```

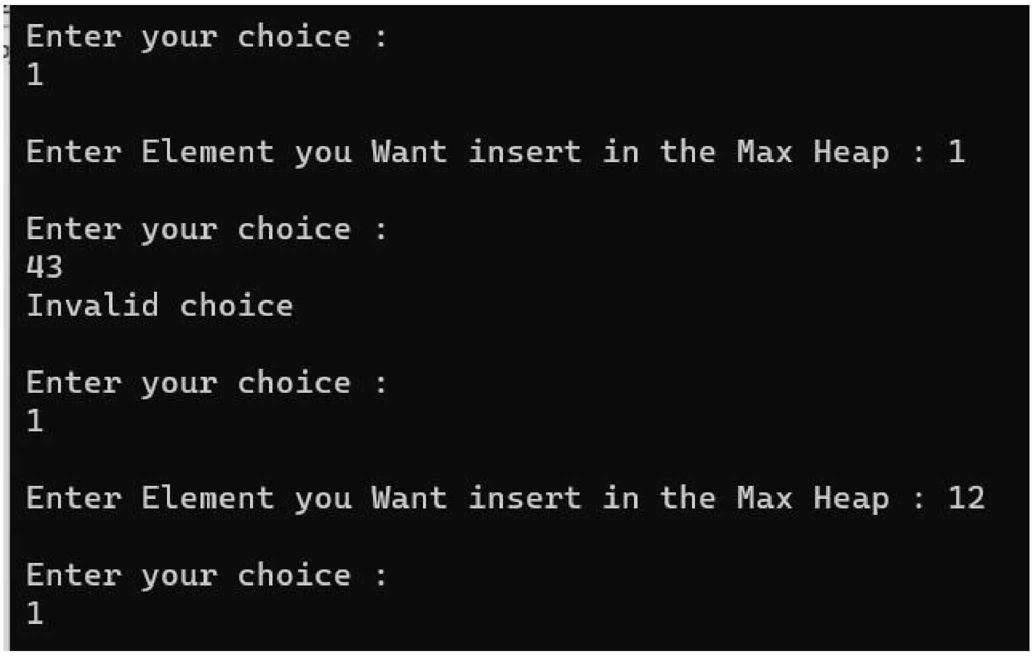
```
			data[heapsize]=val;
			reheapUp(heapsize);
			heapsize++;
		}
	}
	void BinaryMaxHeap::reheapUp(int node)
	{
		int parentIndex=getParentChildIndex(node);
		if(node!=0)
		{
			if(data[parentIndex]<data[node])
			{
				int temp=data[parentIndex];
				data[parentIndex]=data[node];
				data[node]=temp;
				reheapUp(parentIndex);
			}
		}
	}
	void BinaryMaxHeap::remove()
	{
		if(heapsize==0)
		{
			cout<<"\nEmpty Heap";
		}
		else
		{
			cout<<"\n"<<data[0] << " is Removed from the Max Heap.";
			data[0]=heapsize-1;
			reheapDown(0);
			heapsize--;
		}
	}
	void BinaryMaxHeap::reheapDown(int node)
	{
		int tempIndex;
		int Left=getLeftChildIndex(node);
		int Right=getRightChildIndex(node);
		if(Right>=heapsize)
		{
			if(Left>=heapsize)
				return;
			else
				tempIndex=Left;
		}
		else
		{
			if(data[Left]>data[Right])
			{
				tempIndex=Left;
			}
			else
				tempIndex=Right;
		}
		if(data[tempIndex]>data[node])
		{
```

```
                        int temp=data[tempIndex];
                        data[tempIndex]=data[node];
                        data[node]=temp;
                        reheapDown(tempIndex);
            }
}
int BinaryMaxHeap::getMax()
{
            if(heapsize==0)
            {
                        cout<<"Empty Heap";
                        return -1;
            }
            else
                        return data[0];
}
void BinaryMaxHeap::checkSpace(){
                                if(heapsize==0){
                                        cout << "\nHeap is Empty.";
                                }else if(heapsize == arraysize){
                                        cout << "\nSorry to Inform you Heap is Full.";
                                }else{
                                        int k = arraysize - heapsize;
                                        cout << "\nYou can add " << k << " more Elements in
the given heap with their size is " << arraysize;
                                }
}
int main()
{
            int size;
            int k;
            cout << "\nTo Create Max Heap Press 1:";
            cin >> k;
            if (k != 1){
                        return 0;
            }
            cout << "\nEnter size of Heap: ";
    cin >> size;
    BinaryMaxHeap bn(size);
            int ch, p;
            cout << "1) Insert element to Heap: " << endl;
    cout << "2) Delete element from Heap: " << endl;
    cout << "3) Display all the elements of Heap:" << endl;
    cout << "4) Display the Maximum element in Heap: " << endl;
    cout << "5) Check Available Space in the Heap: " << endl;
    cout << "6) Exit" << endl;

    do {
        cout << "\nEnter your choice : " << endl;
        cin >> ch;
        switch (ch) {
        case 1:
            cout << "\nEnter Element you Want insert in the Max Heap : ";
            cin >> p;
            bn.insert(p);
            break;
        case 2:
            cout << "\nBefore Element removed: ";
            bn.display();
            bn.remove();
            cout << "\nAfter Element removed: ";
            bn.display();
            break;
        case 3:
            cout << "\nDisplay Elements in the Max Heap: ";
            bn.display();
            break;
        case 4:
            cout << "\nDisplay Minimum Element in Max Heap: " << bn.getMax();
            break;
        case 5:
            cout << "\nAvailable Space in the Heap is ";
            bn.checkSpace();
            break;
        case 6:
            exit(0);
        default:
            cout << "Invalid choice" << endl;
        }
    } while (ch != 6);
    return 0;
}
```

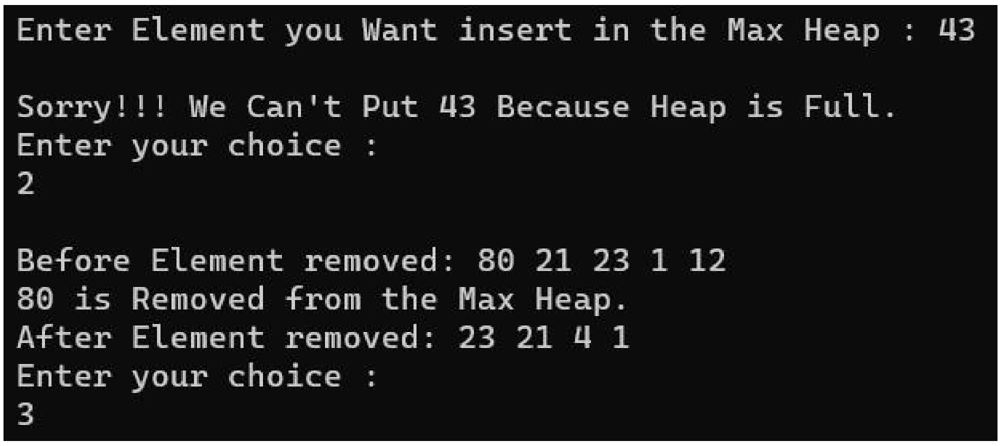
Output:



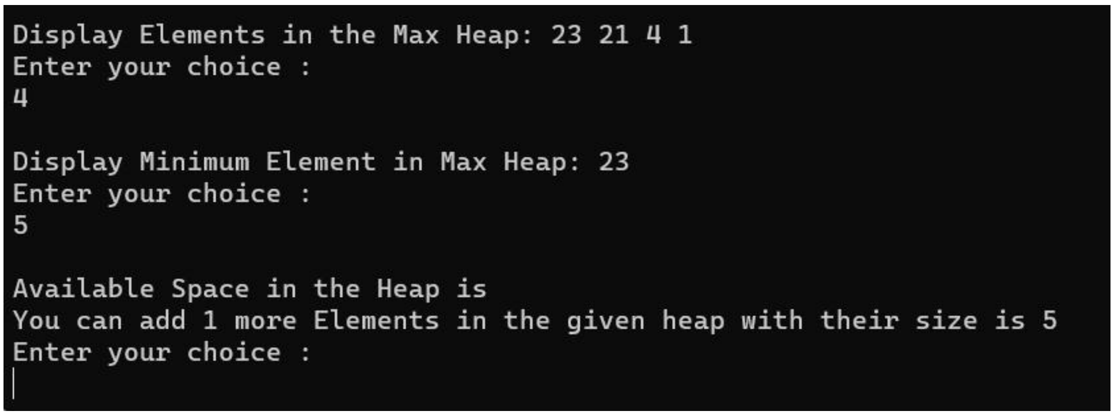
```
To Create Max Heap Press 1:1

Enter size of Heap: 5
1) Insert element to Heap:
2) Delete element from Heap:
3) Display all the elements of Heap:
4) Display the Maximum element in Heap:
5) Check Available Space in the Heap:
6) Exit

Enter your choice :
12
Invalid choice

Enter your choice :
1

Enter Element you Want insert in the Max Heap : 23

Enter your choice :
1

Enter Element you Want insert in the Max Heap : 21

Enter your choice :
1

Enter Element you Want insert in the Max Heap : 80
```



```
Enter your choice :
1

Enter Element you Want insert in the Max Heap : 1

Enter your choice :
43
Invalid choice

Enter your choice :
1

Enter Element you Want insert in the Max Heap : 12

Enter your choice :
1
```



```
Enter Element you Want insert in the Max Heap : 43

Sorry!!! We Can't Put 43 Because Heap is Full.
Enter your choice :
2

Before Element removed: 80 21 23 1 12
80 is Removed from the Max Heap.
After Element removed: 23 21 4 1
Enter your choice :
3
```



```
Display Elements in the Max Heap: 23 21 4 1
Enter your choice :
4

Display Minimum Element in Max Heap: 23
Enter your choice :
5

Available Space in the Heap is
You can add 1 more Elements in the given heap with their size is 5
Enter your choice :
```