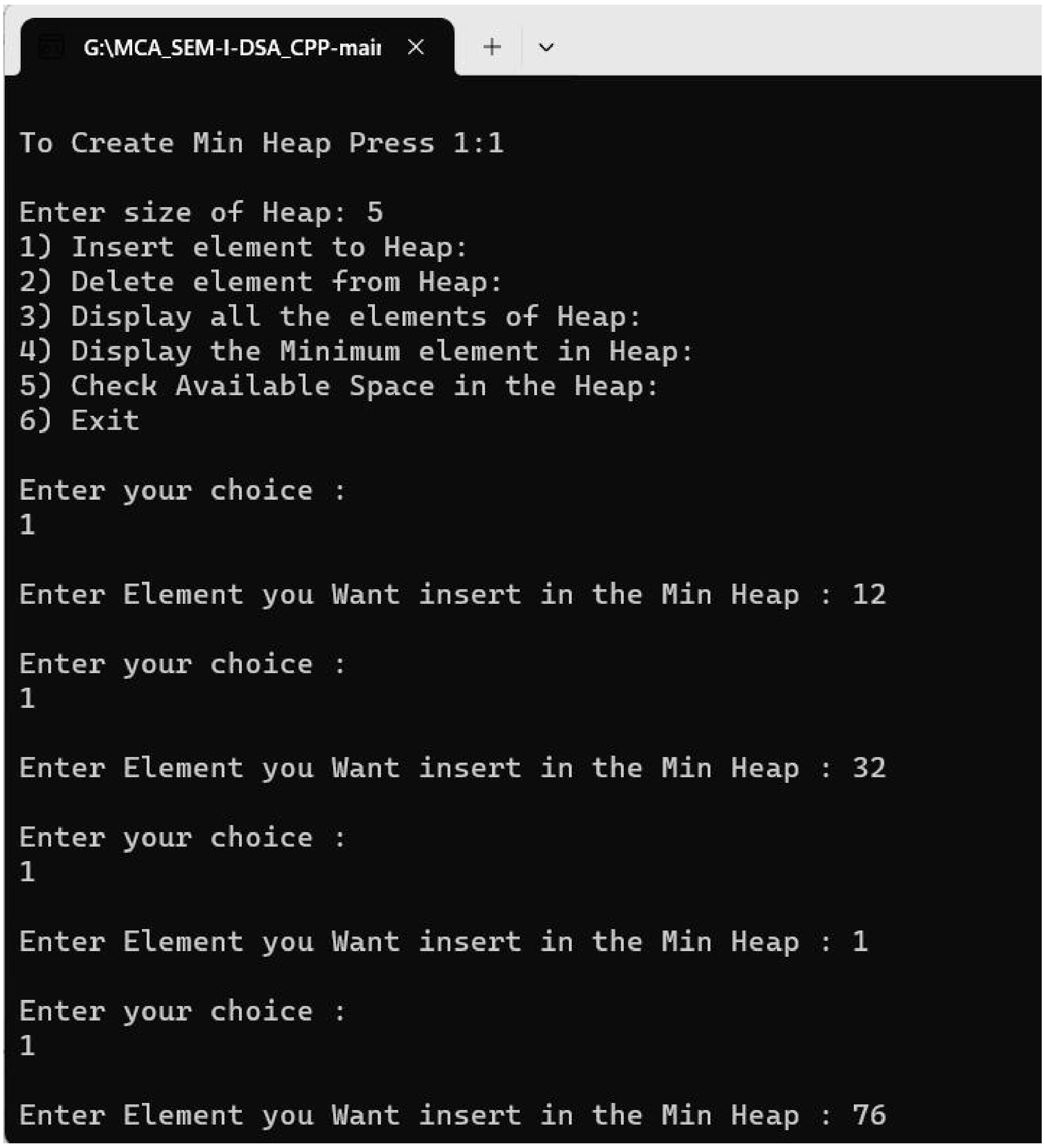
MinHeap.cpp:

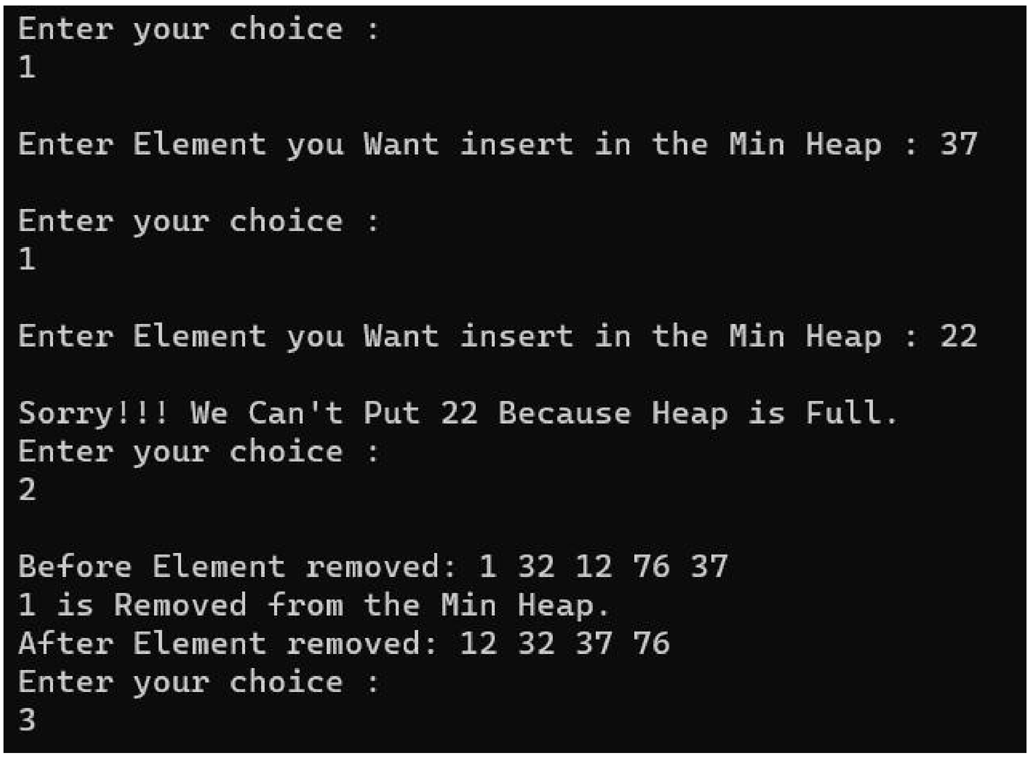
```
#include<iostream>
#include<stdlib.h>
#include<conio.h>

using namespace std;

class BinaryMinHeap
{
        public:
         int *data;
         int heapS;
         int arrS;
         BinaryMinHeap(int size)
         {
                 data=new int[size];
                 heapS=0;
                 arrS=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 reheapDown(int node);
         void checkSpace();
         int getMin();
};
int BinaryMinHeap::getLeftChildIndex(int node)
{
        return((2*node)+1);
}
int BinaryMinHeap::getRightChildIndex(int node)
{
        return((2*node)+2);
}
int BinaryMinHeap::getParentChildIndex(int node)
{
        return((node-1)/2);
}
void BinaryMinHeap::display()
{
        for(int i=0;i<heapS;i++)
         {
                 cout<<data[i]<<" ";
         }
}
void BinaryMinHeap::insert(int val)
{
         if(heapS==arrS)
         {
                cout<<"\nSorry!!! We Can't Put "<< val <<" Because Heap is Full.";
         }
```

```
            else
            {
                        data[heapS]=val;
                        reheapUp(heapS);
                        heapS++;
            }
}
void BinaryMinHeap::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 BinaryMinHeap::remove()
{
            if(heapS==0)
             {
                        cout<<"\nEmpty Heap";
             }
             else
             {
                        cout<<"\n"<<data[0] << " is Removed from the Min Heap.";
                        data[0]=data[heapS-1];
                        reheapDown(0);
                        heapS--;
             }
}
void BinaryMinHeap::reheapDown(int node)
{
            int tempIndex;
             int Left=getLeftChildIndex(node);
             int Right=getRightChildIndex(node);
             if(Right>=heapS)
             {
                        if(Left>=heapS)
                                   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 BinaryMinHeap::getMin()
{
        if(heapS==0)
         {
                cout<<"Empty Heap!!!";
                return 0;
         }
         else
                return data[0];
}
void BinaryMinHeap::checkSpace(){
                        if(heapS==0){
                                cout << "\nHeap is Empty.";
                        }else if(heapS == arrS){
                                cout << "\nSorry to Inform you Heap is Full.";
                        }else{
                                int k = arrS - heapS;
                                cout << "\nYou can add " << k << " more Elements in
the given heap with their size is " << arrS;
                        }
}
int main()
{
        int size;
        int k;
        cout << "\nTo Create Min Heap Press 1:";
        cin >> k;
        if (k != 1){
                return 0;
        }
        cout << "\nEnter size of Heap: ";
   cin >> size;
   BinaryMinHeap 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 Minimum 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 Min 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 Min Heap: ";
            bn.display();
            break;
        case 4:
            cout << "\nDisplay Minimum Element in Min Heap: " << bn.getMin();
            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;
}
```

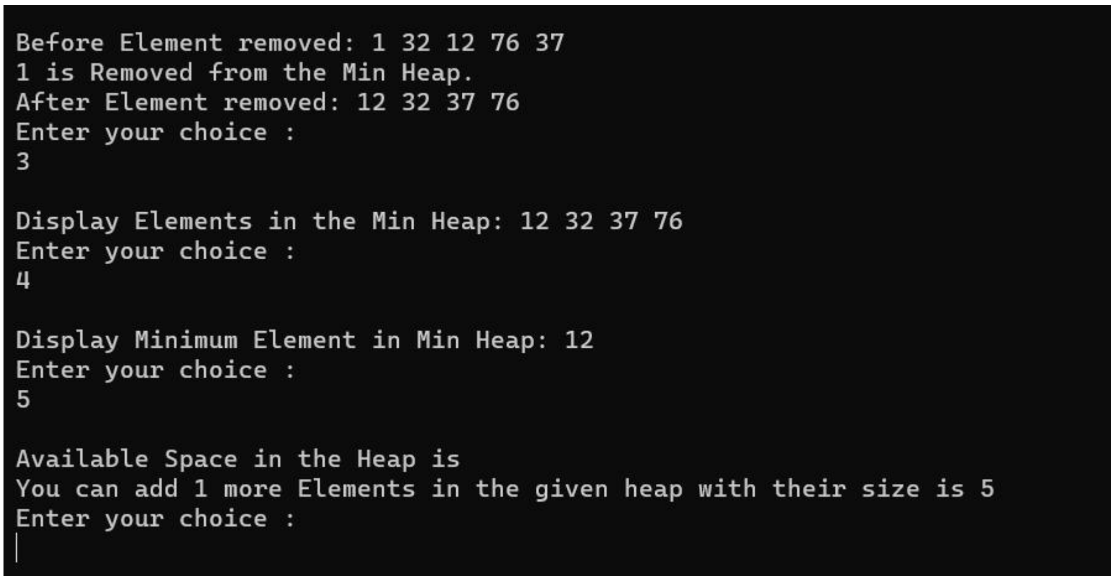
Output:



```
To Create Min 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 Minimum element in Heap:
5) Check Available Space in the Heap:
6) Exit

Enter your choice :
1

Enter Element you Want insert in the Min Heap : 12

Enter your choice :
1

Enter Element you Want insert in the Min Heap : 32

Enter your choice :
1

Enter Element you Want insert in the Min Heap : 1

Enter your choice :
1

Enter Element you Want insert in the Min Heap : 76
```



```
Enter your choice :
1

Enter Element you Want insert in the Min Heap : 37

Enter your choice :
1

Enter Element you Want insert in the Min Heap : 22

Sorry!!! We Can't Put 22 Because Heap is Full.
Enter your choice :
2

Before Element removed: 1 32 12 76 37
1 is Removed from the Min Heap.
After Element removed: 12 32 37 76
Enter your choice :
3
```



```
Before Element removed: 1 32 12 76 37
1 is Removed from the Min Heap.
After Element removed: 12 32 37 76
Enter your choice :
3

Display Elements in the Min Heap: 12 32 37 76
Enter your choice :
4

Display Minimum Element in Min Heap: 12
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 :
```