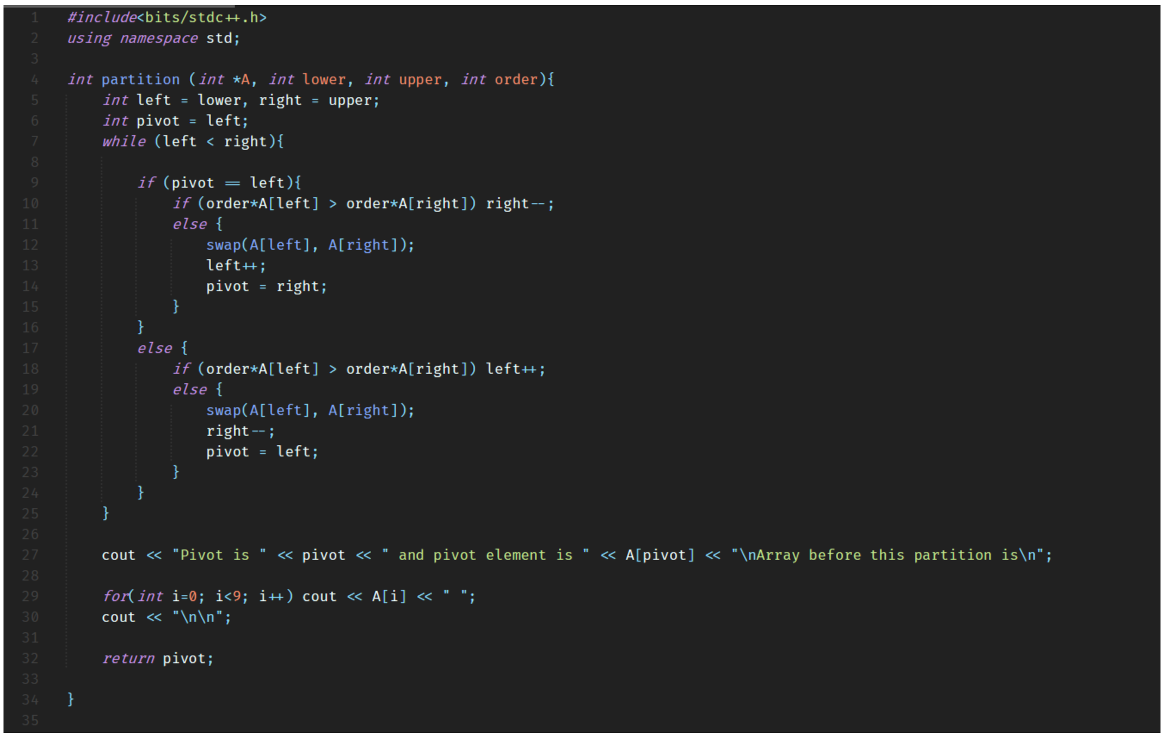
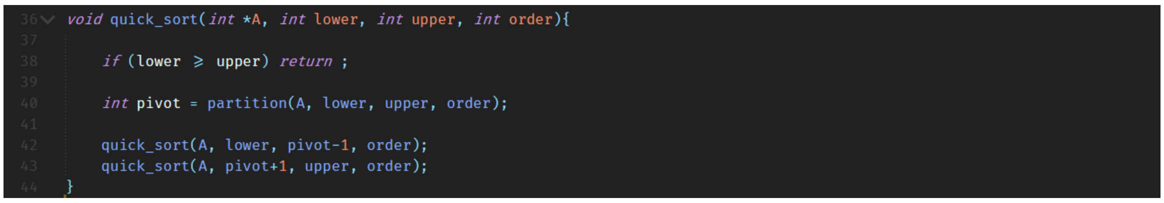
Harsh_Kumar_MAN_106_Assignment_7

🌴 Note : The order of questions is : 2 1 3 5 4

- **Answer 2:**
  - Program :



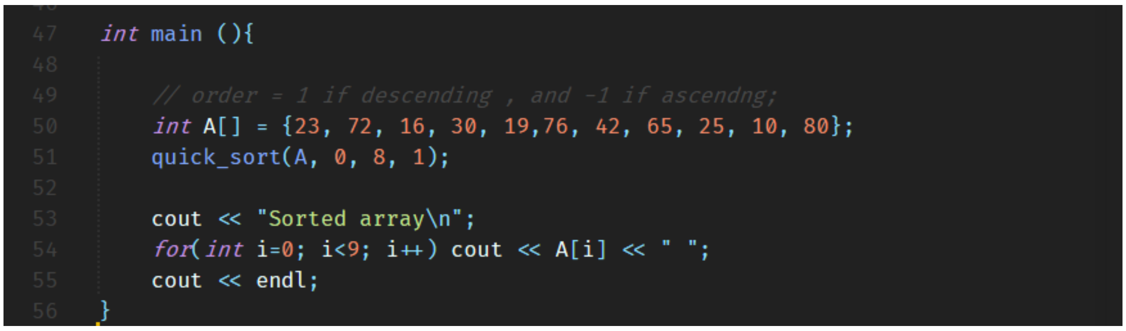
```cpp
#include<bits/stdc++.h>
using namespace std;

int partition (int *A, int lower, int upper, int order){
    int left = lower, right = upper;
    int pivot = left;
    while (left < right){

        if (pivot == left){
            if (order*A[left] > order*A[right]) right--;
            else {
                swap(A[left], A[right]);
                left++;
                pivot = right;
            }
        }
        else {
            if (order*A[left] > order*A[right]) left++;
            else {
                swap(A[left], A[right]);
                right--;
                pivot = left;
            }
        }
    }

    cout << "Pivot is " << pivot << " and pivot element is " << A[pivot] << "\nArray before this partition is\n";

    for(int i=0; i<9; i++) cout << A[i] << " ";
    cout << "\n\n";

    return pivot;

}

```



```cpp
void quick_sort(int *A, int lower, int upper, int order){

    if (lower >= upper) return ;

    int pivot = partition(A, lower, upper, order);

    quick_sort(A, lower, pivot-1, order);
    quick_sort(A, pivot+1, upper, order);
}
```

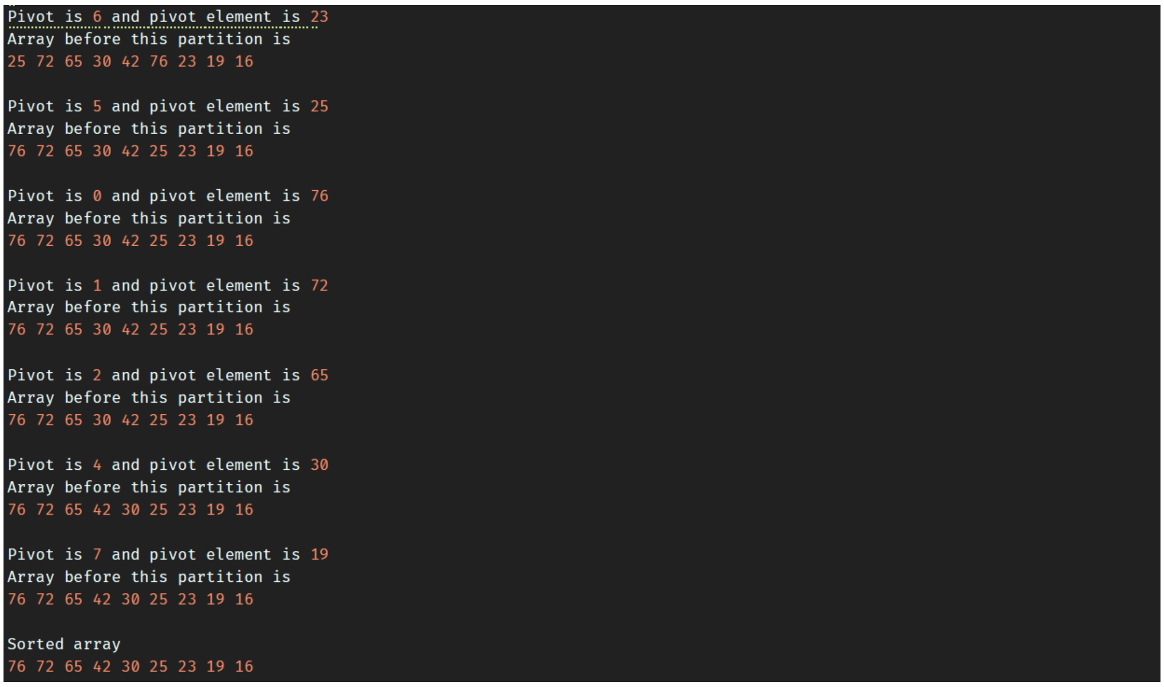
- **Answer 1:**
  - Using the above program we will sort following sequence of keys to arrange them in descending order :

23, 72, 16, 30, 19,76, 42, 65, 25, 10, 80



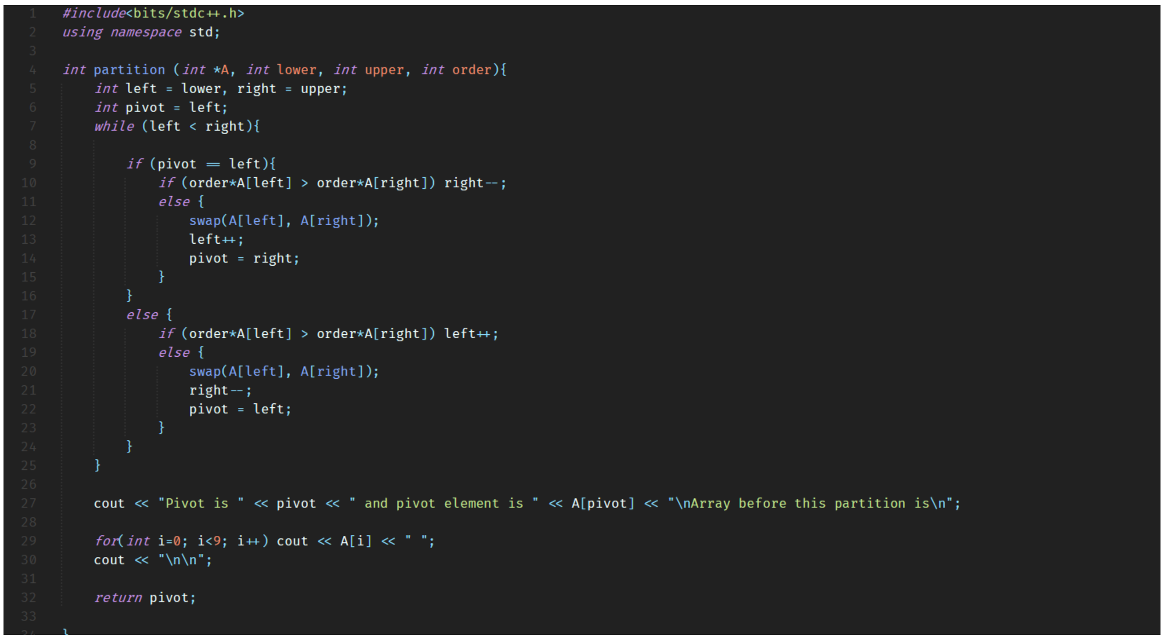
```cpp
int main (){

    // order = 1 if descending , and -1 if ascendng;
    int A[] = {23, 72, 16, 30, 19,76, 42, 65, 25, 10, 80};
    quick_sort(A, 0, 8, 1);

    cout << "Sorted array\n";
    for( int i=0; i<9; i++) cout << A[i] << " ";
    cout << endl;
}
```

- Output :

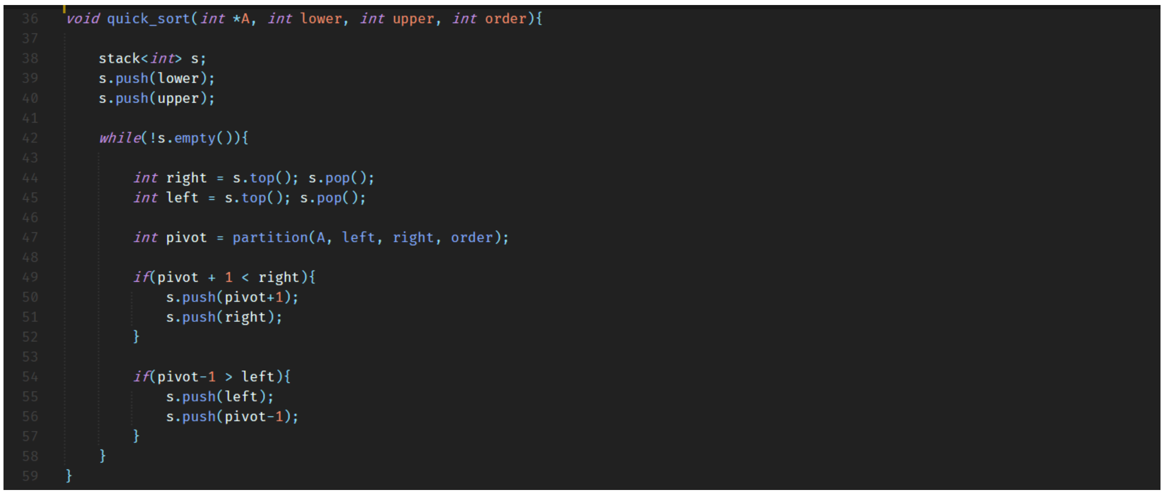


```
Pivot is 6 and pivot element is 23
Array before this partition is
25 72 65 30 42 76 23 19 16

Pivot is 5 and pivot element is 25
Array before this partition is
76 72 65 30 42 25 23 19 16

Pivot is 0 and pivot element is 76
Array before this partition is
76 72 65 30 42 25 23 19 16

Pivot is 1 and pivot element is 72
Array before this partition is
76 72 65 30 42 25 23 19 16

Pivot is 2 and pivot element is 65
Array before this partition is
76 72 65 30 42 25 23 19 16

Pivot is 4 and pivot element is 30
Array before this partition is
76 72 65 42 30 25 23 19 16

Pivot is 7 and pivot element is 19
Array before this partition is
76 72 65 42 30 25 23 19 16

Sorted array
76 72 65 42 30 25 23 19 16
```

- **Answer 3:**
  - Program :

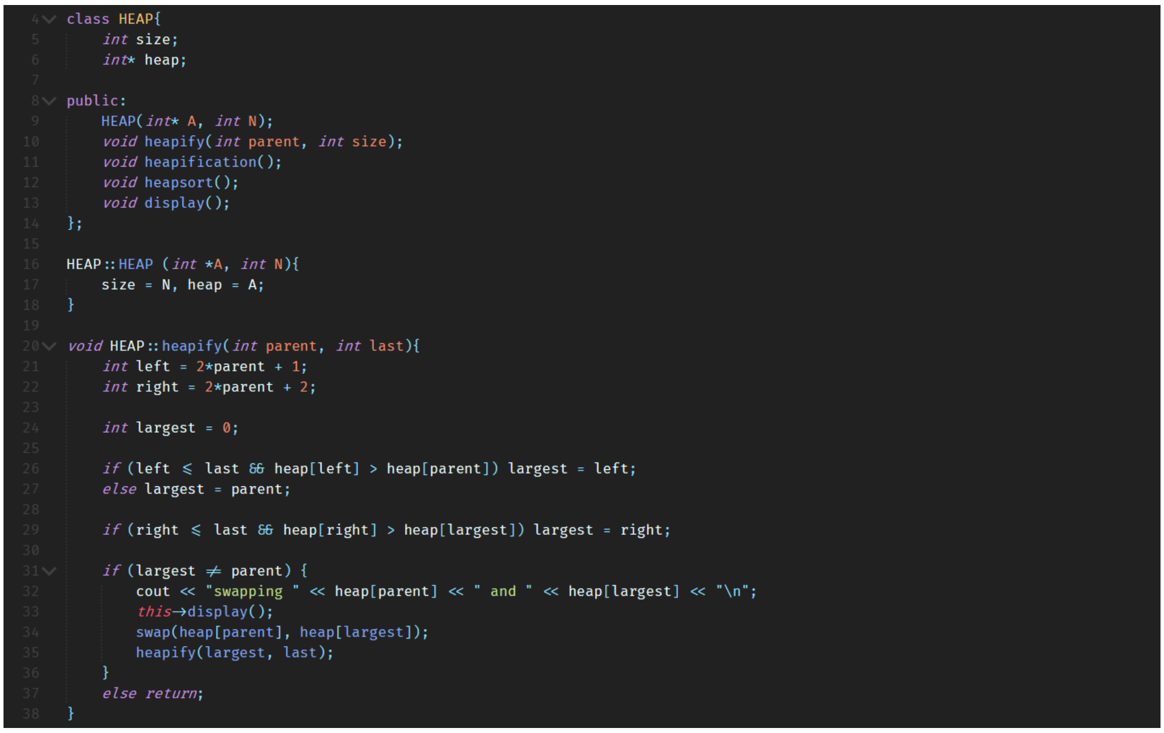


```cpp
#include<bits/stdc++.h>
using namespace std;

int partition (int *A, int lower, int upper, int order){
    int left = lower, right = upper;
    int pivot = left;
    while (left < right){

        if (pivot == left){
            if (order*A[left] > order*A[right]) right--;
            else {
                swap(A[left], A[right]);
                left++;
                pivot = right;
            }
        }
        else {
            if (order*A[left] > order*A[right]) left++;
            else {
                swap(A[left], A[right]);
                right--;
                pivot = left;
            }
        }
    }

    cout << "Pivot is " << pivot << " and pivot element is " << A[pivot] << "\nArray before this partition is\n";

    for( int i=0; i<9; i++) cout << A[i] << " ";
    cout << "\n\n";

    return pivot;

}
```

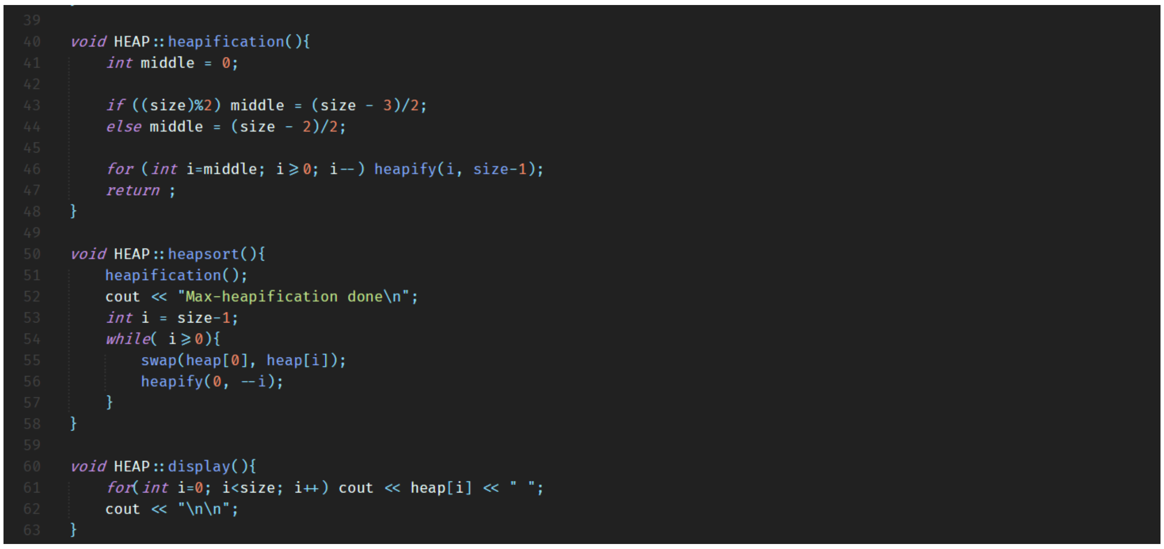


```cpp
void quick_sort(int *A, int lower, int upper, int order){

    stack<int> s;
    s.push(lower);
    s.push(upper);

    while(!s.empty()){

        int right = s.top(); s.pop();
        int left = s.top(); s.pop();

        int pivot = partition(A, left, right, order);

        if(pivot + 1 < right){
            s.push(pivot+1);
            s.push(right);
        }

        if(pivot-1 > left){
            s.push(left);
            s.push(pivot-1);
        }
    }
}
```

- **Answer 5:**
    - Program :



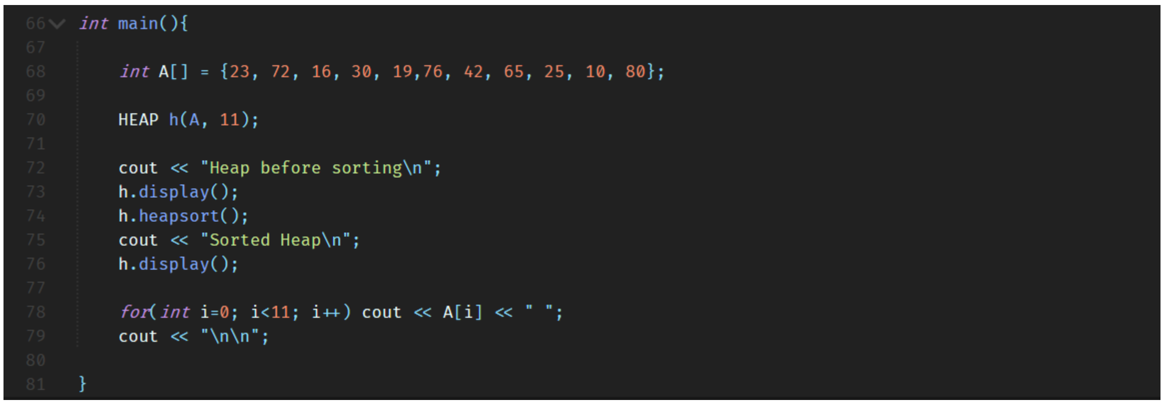
```cpp
class HEAP{
    int size;
    int* heap;

public:
    HEAP(int* A, int N);
    void heapify(int parent, int size);
    void heapification();
    void heapsort();
    void display();
};

HEAP::HEAP (int *A, int N){
    size = N, heap = A;
}

void HEAP::heapify(int parent, int last){
    int left = 2*parent + 1;
    int right = 2*parent + 2;

    int largest = 0;

    if (left <= last && heap[left] > heap[parent]) largest = left;
    else largest = parent;

    if (right <= last && heap[right] > heap[largest]) largest = right;

    if (largest != parent) {
        cout << "swapping " << heap[parent] << " and " << heap[largest] << "\n";
        this->display();
        swap(heap[parent], heap[largest]);
        heapify(largest, last);
    }
    else return;
}
```



```cpp

void HEAP::heapification(){
    int middle = 0;

    if ((size)%2) middle = (size - 3)/2;
    else middle = (size - 2)/2;

    for (int i=middle; i>=0; i--) heapify(i, size-1);
    return ;
}

void HEAP::heapsort(){
    heapification();
    cout << "Max-heapification done\n";
    int i = size-1;
    while( i>=0){
        swap(heap[0], heap[i]);
        heapify(0, --i);
    }
}

void HEAP::display(){
    for(int i=0; i<size; i++) cout << heap[i] << " ";
    cout << "\n\n";
}
```

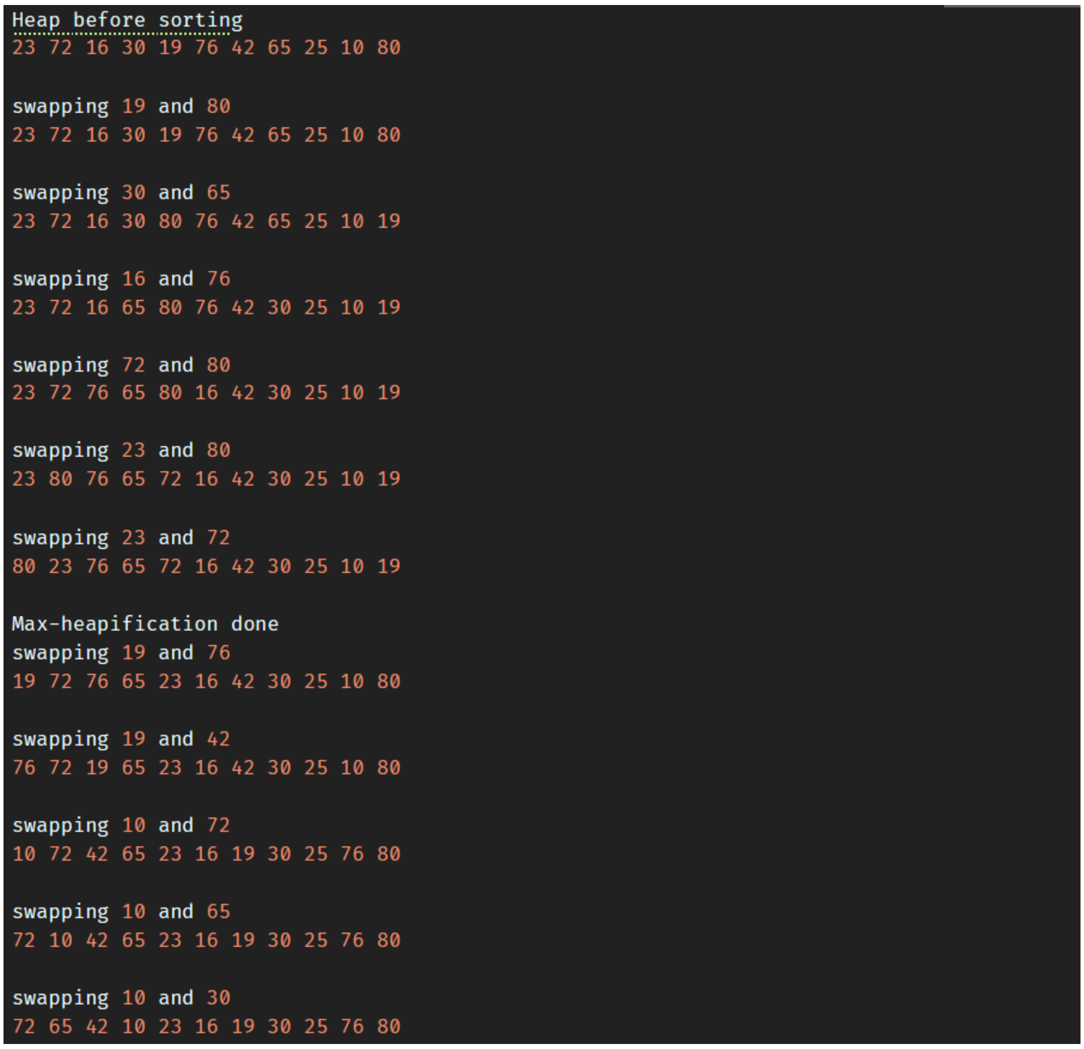
- **Answer 4:**
    - Using the above program we will sort following sequence of keys to arrange them in descending order :

23, 72, 16, 30, 19,76, 42, 65, 25, 10, 80

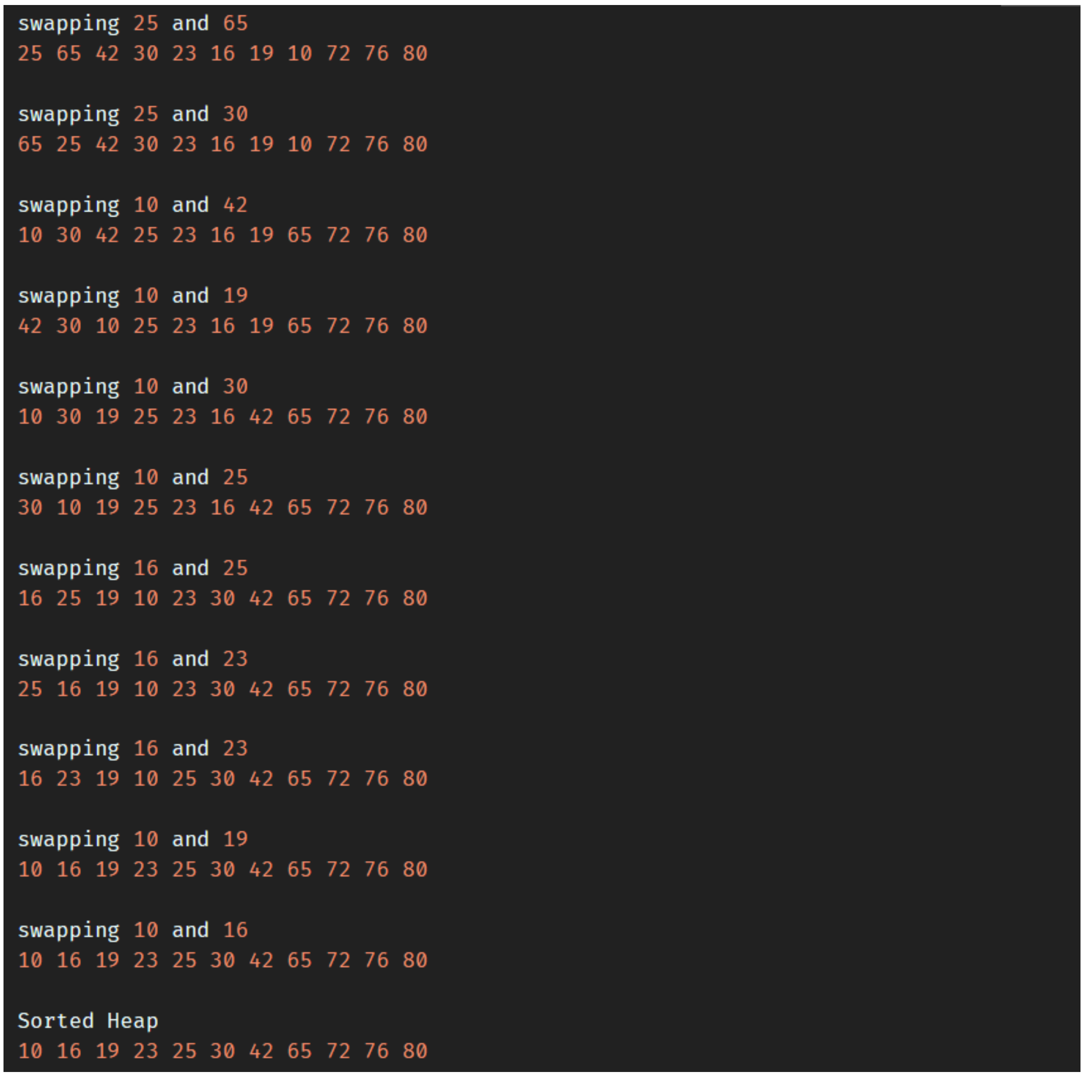


```cpp
int main(){

    int A[] = {23, 72, 16, 30, 19,76, 42, 65, 25, 10, 80};

    HEAP h(A, 11);

    cout << "Heap before sorting\n";
    h.display();
    h.heapsort();
    cout << "Sorted Heap\n";
    h.display();

    for( int i=0; i<11; i++) cout << A[i] << " ";
    cout << "\n\n";

}
```

- Output :



```
Heap before sorting
23 72 16 30 19 76 42 65 25 10 80

swapping 19 and 80
23 72 16 30 19 76 42 65 25 10 80

swapping 30 and 65
23 72 16 30 80 76 42 65 25 10 19

swapping 16 and 76
23 72 16 65 80 76 42 30 25 10 19

swapping 72 and 80
23 72 76 65 80 16 42 30 25 10 19

swapping 23 and 80
23 80 76 65 72 16 42 30 25 10 19

swapping 23 and 72
80 23 76 65 72 16 42 30 25 10 19

Max-heapification done
swapping 19 and 76
19 72 76 65 23 16 42 30 25 10 80

swapping 19 and 42
76 72 19 65 23 16 42 30 25 10 80

swapping 10 and 72
10 72 42 65 23 16 19 30 25 76 80

swapping 10 and 65
72 10 42 65 23 16 19 30 25 76 80

swapping 10 and 30
72 65 42 10 23 16 19 30 25 76 80
```



```
swapping 25 and 65
25 65 42 30 23 16 19 10 72 76 80

swapping 25 and 30
65 25 42 30 23 16 19 10 72 76 80

swapping 10 and 42
10 30 42 25 23 16 19 65 72 76 80

swapping 10 and 19
42 30 10 25 23 16 19 65 72 76 80

swapping 10 and 30
10 30 19 25 23 16 42 65 72 76 80

swapping 10 and 25
30 10 19 25 23 16 42 65 72 76 80

swapping 16 and 25
16 25 19 10 23 30 42 65 72 76 80

swapping 16 and 23
25 16 19 10 23 30 42 65 72 76 80

swapping 16 and 23
16 23 19 10 25 30 42 65 72 76 80

swapping 10 and 19
10 16 19 23 25 30 42 65 72 76 80

swapping 10 and 16
10 16 19 23 25 30 42 65 72 76 80

Sorted Heap
10 16 19 23 25 30 42 65 72 76 80
```