lab_11.cpp

```cpp
// <----Lab 11 - Heap----->

#include <iostream>
#include <vector>
using namespace std;

void swap(int *a, int *b)
{
    int temp = *b;
    *b = *a;
    *a = temp;
}

void heapifyMax(vector<int> &hT, int i, int size)
{
    int largest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;

    if (l < size && hT[l] > hT[largest])
        largest = l;
    if (r < size && hT[r] > hT[largest])
        largest = r;

    if (largest != i)
    {
        swap(&hT[i], &hT[largest]);
        heapifyMax(hT, largest, size);
    }
}

void insertMax(vector<int> &hT, int newNum)
{
    int size = hT.size();
    if (size == 0)
        hT.push_back(newNum);
    else
    {
        hT.push_back(newNum);
        for (int i = size / 2 - 1; i >= 0; i--)
            heapifyMax(hT, i, size);
    }
}

void deleteNodeMax(vector<int> &hT, int num)
{
    int size = hT.size();
    int i;
    for (i = 0; i < size; i++)
    {
        if (num == hT[i])
            break;
    }
```

```
    swap(&hT[i], &hT[size - 1]);
    hT.pop_back();
    for (int i = size / 2 - 1; i >= 0; i--)
        heapifyMax(hT, i, size - 1);
}

// Q1. The above code is for Max Heap. Implement a Min Heap and then insert and delete elements.

void heapifyMin(vector<int> &hT, int i, int size)
{
    int smallest = i;
    int l = 2 * i + 1;
    int r = 2 * i + 2;

    if (l < size && hT[l] < hT[smallest])
        smallest = l;
    if (r < size && hT[r] < hT[smallest])
        smallest = r;

    if (smallest != i)
    {
        swap(&hT[i], &hT[smallest]);
        heapifyMin(hT, smallest, size);
    }
}

void insertMin(vector<int> &hT, int newNum)
{
    int size = hT.size();
    if (size == 0)
        hT.push_back(newNum);
    else
    {
        hT.push_back(newNum);
        int i = size;
        while (i > 0 && hT[(i - 1) / 2] > hT[i])
        {
            swap(&hT[i], &hT[(i - 1) / 2]);
            i = (i - 1) / 2;
        }
    }
}

void deleteNodeMin(vector<int> &hT, int num)
{
    int size = hT.size();
    int i;
    for (i = 0; i < size; i++)
    {
        if (num == hT[i])
            break;
    }
    swap(&hT[i], &hT[size - 1]);
    hT.pop_back();
    for (int i = size / 2 - 1; i >= 0; i--)
```

```cpp
        heapifyMin(hT, i, size - 1);
}

// Q2. Implement peek and extract operations for both max and min values.
int extractMin(vector<int> &hT)
{
    if (hT.size() == 0)
    {
        cout << "Heap is empty\n";
        return -1;
    }

    int minVal = hT[0];
    deleteNodeMin(hT, hT[0]);
    return minVal;
}

int peekMin(const vector<int> &hT)
{
    if (hT.size() == 0)
    {
        cout << "Heap is empty\n";
        return -1;
    }
    return hT[0];
}

int peekMax(const vector<int> &hT)
{
    if (hT.size() == 0)
    {
        cout << "Heap is empty\n";
        return -1;
    }
    return hT[0];
}
int extractMax(vector<int> &hT)
{
    if (hT.size() == 0)
    {
        cout << "Heap is empty\n";
        return -1;
    }

    int maxVal = hT[0];
    deleteNodeMax(hT, hT[0]);
    return maxVal;
}

void printArray(const vector<int> &hT)
{
    for (int i = 0; i < hT.size(); ++i)
        cout << hT[i] << " ";
    cout << "\n";
}

```

```cpp
int main()
{
    vector<int> maxHeap;
    insertMax(maxHeap, 3);
    insertMax(maxHeap, 4);
    insertMax(maxHeap, 9);
    insertMax(maxHeap, 5);
    insertMax(maxHeap, 2);

    cout << "Max-Heap array: ";
    printArray(maxHeap);

    deleteNodeMax(maxHeap, 4);
    cout << "After deleting an element in Max-Heap: ";
    printArray(maxHeap);

    cout << "Max-Heap peek: " << peekMax(maxHeap) << endl;
    cout << "Extract Max value: " << extractMax(maxHeap) << endl;
    cout << "Max-Heap array after extraction: ";
    printArray(maxHeap);

    vector<int> minHeap;
    insertMin(minHeap, 3);
    insertMin(minHeap, 4);
    insertMin(minHeap, 9);
    insertMin(minHeap, 5);
    insertMin(minHeap, 2);

    cout << "\nMin-Heap array: ";
    printArray(minHeap);

    deleteNodeMin(minHeap, 4);
    cout << "After deleting an element in Min-Heap: ";
    printArray(minHeap);

    cout << "Min-Heap peek: " << peekMin(minHeap) << endl;
    cout << "Extract Min value: " << extractMin(minHeap) << endl;
    cout << "Min-Heap array after extraction: ";
    printArray(minHeap);

    return 0;
}

```