```
#include <iostream>
#include <cstring>
#include <cstdlib>
using namespace std;

class DaryHeap {
private:
    int d;
    int heapSize;
    int size;
    int *array;
    int childNum;
public:
    DaryHeap(int length, int childNum) {
        heapSize = 0;
        childNum = d;
        size = 10;
        array = new int[length + 1];
        for (int i = 0; i < length + 1; i++)
            array[i] = -1;
        int j = 0;
        while (array[j] != -1)
            j++;
        this->array = array;
        this->d = childNum;
    }

    bool isEmpty() {
        return heapSize == 0;
    }

    bool isFull() {
        return heapSize == size;
    }


    void insert(int k) {
        if (isFull()) {
            cout << "Queue is full" << endl;
        }
```

Constructor

check whether is empty or not

insert function

```
41          array[heapSize] = k;
42          heapSize++;
43          swim(heapSize - 1);
44      }
45
46
47      int delMax() {
48          if (isEmpty()) {
49              cout << "Queue is empty" << endl;
50              return 0;
51          }
52          int key = array[0];
53          array[0] = array[heapSize - 1];
54          heapSize--;
55          Sink();
56          return key;
57      }
58
59      void swim(int index) {
60          int parentIndex = findparent(index);
61          while (parent_index >= 0 && index > 0) {
62              if (heap[index] > heap[parent_index]) {
63                  int temp = heap[index];
64                  heap[index] = heap[parent_index];
65                  heap[parent_index] = temp;
66                  index = parent_index;
67                  parent_index = findparent(index);
68              } else
69                  break;
70          }
71      }
72
73
74      int findparent(int i) {
75          return (i - 1) / childNum;
76      }
77
78      int kthChild(int i, int k) {
79          return childNnum * i + k;
80      }
```

delete maximum function

calculation

```
81
82      void Sink(int hole) {
83          int child;
84          int tmp = array[hole];
85          for (kthChild(hole, 1) < heapSize; hole = child) {
86              if (array[child] < tmp)
87                  array[hole] = array[child];
88              else
89                  break;
90          }
91          array[hole] = tmp;
92      }
93
94
95      void printHeap() {
96          for (int i = 0; i < heapSize; i++)
97              cout << array[i] << "   ";
98      }
99
100     void doubleArray() {
101
102     }
103
104     void heap() {
105         for (int index = findparent(heapSize - 1); index >= 0; index
    --) {
106             Sink(index);
107         }
108     }
109 };
110
111 int main() {
112     DaryHeap the;
113     the.insert();
114     the.delMax();
115     the.swim();
116     the.Sink();
117     the.printHeap();
118     return 0;
119 }
```

print fimction.

main function