Câu hỏi 8

Không hoàn thành

Chấm điểm của 1,00

Implement functions: **Peek, Pop, Size, Empty, Contains** to a maxHeap. If the function cannot execute, **return -1**.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <cmath>
#include <vector>
#include <algorithm>
using namespace std;
#define SEPARATOR "#<ab@17943918#@>#"
template<class T>
class Heap {
protected:
    T* elements;
    int capacity;
    int count;
public:
    Heap()
    {
        this->capacity = 10;
        this->count = 0;
        this->elements = new T[capacity];
    }
    ~Heap()
    {
        delete[]elements;
    }
    void push(T item);

    bool isEmpty();
    bool contains(T item);
    T peek();
    bool pop();
    int size();

    void printHeap()
    {
        cout << "Max Heap [ ";
        for (int i = 0; i < count; i++)
            cout << elements[i] << " ";
        cout << "]\n";
    }
private:
    void ensureCapacity(int minCapacity);
    void reheapUp(int position);
    void reheapDown(int position);
};
//Your code goes here
```

For example:

Test	Result

Heap<int> maxHeap;
for (int i=0;i<10;i++){
    maxHeap.push(i);
}
cout << maxHeap.size();
	10

Test	Result

`Heap<int> maxHeap;`
`for (int i=0;i<10;i++){`
`    maxHeap.push(i);`
`}`
`cout << maxHeap.isEmpty();`
	0

Answer: (penalty regime: 0 %)

Reset answer

```
template<class T>
int Heap<T>::size(){
}

template<class T>
bool Heap<T>::isEmpty(){
}

template<class T>
T Heap<T>::peek(){
}

template<class T>
bool Heap<T>::contains(T item){
}

template<class T>
bool Heap<T>::pop(){
}
```

Precheck Kiểm tra

Câu hỏi 9

Không hoàn thành

Chấm điểm của 1,00

Implement function push to push a new item to a maxHeap. You also have to implement ensureCapacity and reheapUp to help you achieve that.

```
template
class Heap{
protected:
    T *elements;
    int capacity;
    int count;

public:
    Heap()
    {
        this->capacity = 10;
        this->count = 0;
        this->elements = new T[capacity];
    }
    ~Heap()
    {
        delete []elements;
    }
    void push(T item);
    void printHeap()
    {
        cout << "Max Heap [ ";
        for (int i = 0; i < count; i++)
            cout << elements[i] << " ";
        cout << "]";
    }

private:
    void ensureCapacity(int minCapacity);
    void reheapUp(int position);
};

// Your code here
```

For example:

Test	Result

`Heap<int> maxHeap;`
`for(int i = 0; i <5;i++)`
`    maxHeap.push(i);`
`maxHeap.printHeap();`
	`Max Heap [ 4 3 1 0 2 ]`

Answer: (penalty regime: 0 %)

Reset answer

```
template<class T>
void Heap<T>::push(T item){

}

template<class T>
void Heap<T>::ensureCapacity(int minCapacity){

}

template<class T>
void Heap<T>::reheapUp(int position){

}
```

Precheck   Kiểm tra

Câu hỏi 10

Không hoàn thành

Chấm điểm của 1,00

Given an array which the elements in it are random. Now we want to build a Max heap from this array. Implement functions Reheap up and Reheap down to heapify element at index position. We will use it to build a heap in next question.

To keep things simple, this question will separate the heap array, not store it in the class heap

```
void reheapDown(int maxHeap[], int numberOfElements, int index);
void reheapUp(int maxHeap[], int numberOfElements, int index);
```

For example:

Test	Result

`int arr[] = {1,2,3,4,5,6,7,8};`
`int size = sizeof(arr)/sizeof(arr[0]);`
`reheapDown(arr,size,0);`
`cout << "[ ";`
`for(int i=0;i<size;i++)`
`    cout << arr[i] << " ";`
`cout << "]";`
	`[ 3 2 7 4 5 6 1 8 ]`

`int arr[] = {1,2,3,4,5,6,7,8};`
`int size = sizeof(arr)/sizeof(arr[0]);`
`reheapUp(arr,size,7);`
`cout << "[ ";`
`for(int i=0;i<size;i++)`
`    cout << arr[i] << " ";`
`cout << "]";`
	`[ 8 1 3 2 5 6 7 4 ]`

Answer: (penalty regime: 0 %)

Reset answer

```
void reheapDown(int maxHeap[], int numberOfElements, int index)
{

}

void reheapUp(int maxHeap[], int numberOfElements, int index)
{

}
```

Precheck Kiểm tra

Câu hỏi 11

Không hoàn thành

Chấm điểm của 1,00

Implement method remove to **remove** the element with given value from a **maxHeap**, **clear** to remove all elements and bring the heap back to the initial state. You also have to implement method **getItem** to help you. Some given methods that you don't need to implement again are **push**, **printHeap**, **ensureCapacity**, **reheapUp**, **reheapDown**.

```
class Heap {
protected:
    T* elements;
    int capacity;
    int count;
public:
    Heap()
    {
        this->capacity = 10;
        this->count = 0;
        this->elements = new T[capacity];
    }
    ~Heap()
    {
        delete[]elements;
    }
    void push(T item);
    int getItem(T item);
    void remove(T item);
    void clear();
    void printHeap()
    {
        cout << "Max Heap [ ";
        for (int i = 0; i < count; i++)
            cout << elements[i] << " ";
        cout << "]\n";
    }
private:
    void ensureCapacity(int minCapacity);
    void reheapUp(int position);
    void reheapDown(int position);
};

// Your code here
```

For example:

Test	Result

`Heap<int> maxHeap;`
`int arr[] = {42,35,30,15,20,21,18,3,7,14};`
`for (int i = 0; i < 10; i++)`
`    maxHeap.push(arr[i]);`
`maxHeap.remove(42);`
`maxHeap.remove(35);`
`maxHeap.remove(30);`
`maxHeap.printHeap();`
	`Max Heap [ 21 20 18 15 14 7 3 ]`

`Heap<int> maxHeap;`
`int arr[] = {78, 67, 32, 56, 8, 23, 19, 45};`
`for (int i = 0; i < 8; i++)`
`    maxHeap.push(arr[i]);`
`maxHeap.remove(78);`
`maxHeap.printHeap();`
	`Max Heap [ 67 56 32 45 8 23 19 ]`

Test	Result

`Heap<int> maxHeap;`
`int arr[] = { 13, 19, 20, 7, 15, 12, 16, 10, 8, 9, 3, 6, 18, 2, 14, 1, 17, 4, 11, 5 };`
`for (int i = 0; i < 20; ++i)`
`    maxHeap.push(arr[i]);`
`maxHeap.clear();`
`maxHeap.printHeap();`
	`Max Heap [ ]`

Answer: (penalty regime: 0 %)

Reset answer

```
template<class T>
int Heap<T>::getItem(T item) {
    // TODO: return the index of item in heap

}

template<class T>
void Heap<T>::remove(T item) {
    // TODO: remove the element with value equal to item

}

template<class T>
void Heap<T>::clear() {
    // TODO: delete all elements in heap

}
```

Precheck Kiểm tra

Câu hỏi 12

Không hoàn thành

Chấm điểm của 1,00

Your task is to implement heap sort (in ascending order) on an unsorted array.

```
#define SEPARATOR "#<ab@17943918#@>#"
#ifndef SORTING_H
#define SORTING_H
#include <iostream>
#include <queue>
using namespace std;
template <class T>
class Sorting {
public:
    /* Function to print an array */
    static void printArray(T *start, T *end)
    {
        long size = end - start;
        for (int i = 0; i < size - 1; i++)
            cout << start[i] << ", ";
        cout << start[size - 1];
        cout << endl;
    }

    //Helping functions go here
    static void heapSort(T* start, T* end){
        //TODO
        Sorting<T>::printArray(start,end);
    }

};
#endif /* SORTING_H */
```

For example:

Test	Result

`int arr[4]={4,2,9,1};`
`Sorting<int>::heapSort(&arr[0],&arr[4]);`
	`1, 2, 4, 9`

`int arr[4]={-1,0,2,3};`
`Sorting<int>::heapSort(&arr[0],&arr[4]);`
	`-1, 0, 2, 3`

Answer: (penalty regime: 0 %)

Reset answer

```
//Helping functions go here
static void heapSort(T* start, T* end){
    //TODO
    Sorting<T>::printArray(start,end);
}
```

Precheck Kiểm tra

Câu hỏi 13

Không hoàn thành

Chấm điểm của 1,00

In a fast food restaurant, a customer is served by following the first-come, first-served rule. The manager wants to minimize the total waiting time of his customers. So he gets to decide who is served first, regardless of how sooner or later a person comes.

Different kinds of food take different amounts of time to cook. And he can't cook food for two customers at the same time, which means when he start cooking for customer A, he has to finish A 's order before cooking for customer B. For example, if there are 3 customers and they come at time 0, 1, 2 respectively, the time needed to cook their food is 3, 9, 6 respectively. If the manager uses first-come, first-served rule to serve his customer, the total waiting time will be 3 + 11 + 16 = 30. In case the manager serves his customer in order 1, 3, 2, the total waiting time will be 3 + 7 + 17 = 27.

Note: The manager does not know about the future orders.

In this question, you should implement function **minWaitingTime** to help the customer find minimum total waiting time to serve all his customers. You are also encouraged to use data structure **Heap** to complete this question. You can use your own code of **Heap**, or use functions related to Heap in library <algorithm>.

For example:

Test	Result

`int n = 3;`
`int arrvalTime[] = { 0, 1, 2 };`
`int completeTime[] = { 3, 9, 6 };`
 	
`cout << minWaitingTime(n, arrvalTime, completeTime);`
	27

`int n = 4;`
`int arrvalTime[] = { 0, 4, 2, 5 };`
`int completeTime[] = { 4, 2, 3, 4 };`
 	
`cout << minWaitingTime(n, arrvalTime, completeTime);`
	21

Answer: (penalty regime: 0 %)

Reset answer

```cpp
int minWaitingTime(int n, int arrvalTime[], int completeTime[]) {
    // YOUR CODE HERE
}
```

Precheck Kiểm tra

Câu hỏi 14

Không hoàn thành

Chấm điểm của 1,00

Given the template for class PrinterQueue and 2 methods:

```
1. addNewRequest(int priority, string fileName)

Add a file to queue with priority and fileName. The test cases have maximum of 100 files.
```

```
2. print()
Print the highest priority file's fileName. If there is no file in the queue, print "No file to print";
```

- The files that have the same priority will be printed in FIFO (First In First Out) order

You can implement any additional method.

For example:

Test	Result

`PrinterQueue* myPrinterQueue = new PrinterQueue();`
`myPrinterQueue->addNewRequest(1, "hello.pdf");`
`myPrinterQueue->addNewRequest(2, "goodbye.pdf");`
`myPrinterQueue->addNewRequest(2, "goodnight.pdf");`
`myPrinterQueue->print();`
`myPrinterQueue->print();`
`myPrinterQueue->print();`
	`goodbye.pdf`
	`goodnight.pdf`
	`hello.pdf`

`PrinterQueue* myPrinterQueue = new PrinterQueue();`
`myPrinterQueue->addNewRequest(1, "hello.pdf");`
`myPrinterQueue->print();`
`myPrinterQueue->print();`
`myPrinterQueue->print();`
	`hello.pdf`
	`No file to print`
	`No file to print`

Answer: (penalty regime: 0 %)

Reset answer

```
class PrinterQueue
{
    // your attributes
public:
    // your methods

    void addNewRequest(int priority, string fileName)
    {
        // your code here
    }

    void print()
    {
        // your code here
        // After some logic code, you have to print fileName with endline
    }
};
```

Kiểm tra