

Đã bắt đầu vào lúc	Thứ năm, 30 Tháng mười một 2023, 9:08 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ năm, 30 Tháng mười một 2023, 9:11 PM
Thời gian thực hiện	3 phút 16 giây
Điểm	7,00/7,00
Điểm	10,00 của 10,00 (100%)

Câu hỏi 1

Chính xác

Điểm 1,00 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
<pre>Heap<int> maxHeap; for (int i=0;i<10;i++){ maxHeap.push(i); } cout << maxHeap.size();</pre>	10

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

Answer: (penalty regime: 0, 0, 5, ... %)

Reset answer

```

1  template<class T>
2  int Heap<T>::size(){
3      return this->count;
4  }
5
6  template<class T>
7  bool Heap<T>::isEmpty(){
8      return !this->count ? true : false;
9  }
10
11 template<class T>
12 T Heap<T>::peek(){
13     if(this->isEmpty()) return -1;
14     return elements[0];
15 }
16
17 template<class T>
18 bool Heap<T>::contains(T item){
19     if(this->isEmpty()) return false;
20     for(int i = 0; i < this->count; i++) {
21         if(item == this->elements[i]) return true;
22     }
23     return false;
24 }
25
26 template<class T>
27 bool Heap<T>::pop() {
28     if(this->isEmpty()) return false;
29     this->elements[0] = this->elements[this->count - 1];
30     this->count--;
31     this->reheapDown(0);
32     return true;
33 }

```

	Test	Expected	Got	
✓	<pre> Heap<int> maxHeap; for (int i=0;i<10;i++){ maxHeap.push(i); } cout << maxHeap.size(); </pre>	10	10	✓
✓	<pre> Heap<int> maxHeap; for (int i=0;i<10;i++){ maxHeap.push(i); } cout << maxHeap.isEmpty(); </pre>	0	0	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 2

Chính xác

Điểm 1,00 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 << "]\n";
    }

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, 0, 5, ... %)

Reset answer

```
1 template<class T>
2 void Heap<T>::push(T item){
3     if(this->count >= this->capacity) this->ensureCapacity(this->count);
4     this->elements[this->count] = item;
5     this->count++;
6     this->reheapUp(this->count-1);
7 }
8
```

```

9  template<class T>
10 void Heap<T>::ensureCapacity(int minCapacity){
11     this->capacity = minCapacity * 2;
12     T* newCapacity = new T[capacity];
13     for(int i = 0; i < this->count; i++) newCapacity[i] = this->elements[i];
14     delete [] elements;
15     this->elements = newCapacity;
16 }
17
18 template<class T>
19 void Heap<T>::reheapUp(int position){
20     if(position <= 0) return;
21     int parent = (position - 1)/2;
22     if(elements[position] >= elements[parent]) {
23         int temp = elements[parent];
24         elements[parent] = elements[position];
25         elements[position] = temp;
26         reheapUp(parent);
27     }
28 }

```

	Test	Expected	Got	
✓	<pre> Heap<int> maxHeap; for(int i = 0; i < 5; i++) maxHeap.push(i); maxHeap.printHeap(); </pre>	Max Heap [4 3 1 0 2]	Max Heap [4 3 1 0 2]	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 3

Chính xác

Điểm 1,00 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
<pre>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 << "]";</pre>	[3 2 7 4 5 6 1 8]
<pre>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 << "]";</pre>	[8 1 3 2 5 6 7 4]

Answer: (penalty regime: 0, 0, 5, ... %)

Reset answer

```
1 void reheapDown(int maxHeap[], int numberOfElements, int index)
2 {
3     int lc = 2*index + 1;
4     int rc = 2*index + 2;
5     while(index < numberOfElements && lc < numberOfElements && rc < numberOfElements) {
6         if(maxHeap[index] < maxHeap[lc] || maxHeap[index] < maxHeap[rc]) {
7             int temp = maxHeap[index];
8             if(maxHeap[lc] > maxHeap[rc]) {
9                 maxHeap[index] = maxHeap[lc];
10                maxHeap[lc] = temp;
11                index = lc;
12                lc = 2*index + 1; rc = 2*index + 2;
13            }
14            else {
15                maxHeap[index] = maxHeap[rc];
16                maxHeap[rc] = temp;
17                index = rc;
18                lc = 2*index + 1; rc = 2*index + 2;
19            }
20        }
21        else return;
22    }
23 }
24
25 void reheapUp(int maxHeap[], int numberOfElements, int index)
26 {
27     if(index <= 0) return;
```

```

28         int parent = (index - 1) / 2;
29         if(maxHeap[index] >= maxHeap[parent]) {
30             int temp = maxHeap[parent];
31             maxHeap[parent] = maxHeap[index];
32             maxHeap[index] = temp;
33             reheapUp(maxHeap, numberOfElements, parent);
34         }
35     }

```

	Test	Expected	Got	
✓	<pre> 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 << "]" </pre>	[3 2 7 4 5 6 1 8]	[3 2 7 4 5 6 1 8]	✓
✓	<pre> 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 << "]" </pre>	[8 1 3 2 5 6 7 4]	[8 1 3 2 5 6 7 4]	✓
✓	<pre> int arr[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}; int size = sizeof(arr)/sizeof(arr[0]); reheapUp(arr,size,13); reheapUp(arr,size,12); cout << "["; for(int i=0;i<size;i++) cout << arr[i] << " "; cout << "]" </pre>	[14 2 13 4 5 1 3 8 9 10 11 12 6 7 15]	[14 2 13 4 5 1 3 8 9 10 11 12 6 7 15]	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 4

Chính xác

Điểm 1,00 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
<pre>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();</pre>	Max Heap [21 20 18 15 14 7 3]
<pre>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();</pre>	Max Heap [67 56 32 45 8 23 19]

Test	Result
<pre> 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(); </pre>	Max Heap []

Answer: (penalty regime: 10, 20, 30, ... %)

Reset answer

```

1  template<class T>
2  int Heap<T>::getItem(T item) {
3      // TODO: return the index of item in heap
4      for(int i = 0; i < this->count ; i++) {
5          if(this->elements[i] == item) return i;
6      }
7      return -1;
8  }
9
10 template<class T>
11 void Heap<T>::remove(T item) {
12     // TODO: remove the element with value equal to item
13     int index = this->getItem(item);
14     if(index == -1) return;
15     else {
16         this->elements[index] = this->elements[this->count - 1];
17         --this->count;
18         this->reheapDown(index);
19         this->reheapUp(index);
20     }
21 }
22
23 template<class T>
24 void Heap<T>::clear() {
25     // TODO: delete all elements in heap
26     // then reallocate memory as initial state
27     delete[] this->elements;
28     this->capacity = 10;
29     this->elements = new T[capacity];
30     this->count = 0;
31 }

```

	Test	Expected	Got	
✓	<pre> 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(); </pre>	Max Heap [21 20 18 15 14 7 3]	Max Heap [21 20 18 15 14 7 3]	✓

	Test	Expected	Got	
✓	<pre> 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(); </pre>	<pre> Max Heap [67 56 32 45 8 23 19] </pre>	<pre> Max Heap [67 56 32 45 8 23 19] </pre>	✓
✓	<pre> 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(); </pre>	<pre> Max Heap [] </pre>	<pre> Max Heap [] </pre>	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 5

Chính xác

Điểm 1,00 của 1,00

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

```
#define SEPARATOR "<#<ab@17943918#@>#"
#ifdef 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, 0, 5, ... %)

Reset answer

```
1 //Helping functions go here
2 static void heapify(T* start, T* end, int n) {
3     int size = end - start;
4     int largest = n;
5     int l = 2*n + 1;
6     int r = 2*n + 2;
7     if(l < size && start[l] > start[largest]) largest = l;
8     if(r < size && start[r] > start[largest]) largest = r;
9     if(largest != n) {
10         swap(start[n], start[largest]);
11         heapify(start, end, largest);
12     }
13 }
14 static void heapSort(T* start, T* end){
15     //TODO
16     int size = end - start;
```

```

17     for(int i = size/2-1; i >= 0; i--) heapify(start, end, i);
18     for(int i = size-1; i > 0; i--) {
19         swap(start[0], start[i]);
20         heapify(start, start + i, 0);
21     }
22     Sorting<T>::printArray(start,end);
23
24

```

	Test	Expected	Got	
✓	int arr[4]={4,2,9,1}; Sorting<int>::heapSort(&arr[0],&arr[4]);	1, 2, 4, 9	1, 2, 4, 9	✓
✓	int arr[4]={-1,0,2,3}; Sorting<int>::heapSort(&arr[0],&arr[4]);	-1, 0, 2, 3	-1, 0, 2, 3	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 6

Chính xác

Điểm 1,00 của 1,00

Cho template của class PrinterQueue có 2 phương thức bắt buộc:

1. addNewRequest(int priority, string fileName)

Phương thức đầu tiên sẽ thêm 1 file vào danh sách hàng đợi của máy in (bao gồm độ ưu tiên và tên file). Test case sẽ có tối đa 100 file cùng lúc trong hàng đợi

2. print()

Phương thức thứ hai sẽ in tên file kèm xuống dòng và xóa nó ra khỏi hàng đợi. Nếu không có file nào trong hàng đợi, phương thức sẽ in ra "No file to print" kèm xuống dòng.

PrinterQueue tuân theo các quy tắc sau:

- fileName có độ ưu tiên cao nhất sẽ được in trước.
- Các fileName có cùng độ ưu tiên sẽ in theo thứ tự FIFO (First In First Out) order.

Nhiệm vụ của bạn là hiện thực class PrinterQueue thỏa mãn các yêu cầu dữ liệu trên

Lưu ý: Bạn có thể thay đổi mọi thứ, thêm thư viện cần thiết ngoại trừ thay đổi tên class, prototype của 2 public method bắt buộc.

Giải thích testcase 1: File goodbye.pdf có độ ưu tiên là 2 và được thêm vào sớm hơn file goodnight.pdf (độ ưu tiên = 2) nên sẽ được in trước, sau đó đến file goodnight.pdf và cuối cùng là hello.pdf có độ ưu tiên thấp nhất (1)

For example:

Test	Result
<pre>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();</pre>	<pre>goodbye.pdf goodnight.pdf hello.pdf</pre>
<pre>PrinterQueue* myPrinterQueue = new PrinterQueue(); myPrinterQueue->addNewRequest(1, "hello.pdf"); myPrinterQueue->print(); myPrinterQueue->print(); myPrinterQueue->print();</pre>	<pre>hello.pdf No file to print No file to print</pre>

Answer: (penalty regime: 0, 0, 0, 100 %)

Reset answer

```

1 #include <queue>
2 class PrinterQueue
3 {
4     private:
5         // your attributes
6         struct Req {
7             int priority;
8             string fileName;
9             int order;
10         bool operator < (const Req& r) const {
11             if(priority != r.priority) return priority < r.priority;
12             return order > r.order;
13         }
14     };
15     priority_queue<Req> pq;
16     int order = 0;
17 public:
```

```

18 // your methods
19 void addNewRequest(int priority, string fileName)
20 {
21     // your code here
22     pq.push({priority, fileName, order++});
23 }
24 void print()
25 {
26     // your code here
27     // After some logic code, you have to print fileName with endl
28     if (pq.empty()) cout << "No file to print\n";
29     else {
30         cout << pq.top().fileName << "\n";
31         pq.pop();
32     }
33 }
34 };

```

	Test	Expected	Got	
✓	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	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	hello.pdf No file to print No file to print	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 7

Chính xác

Điểm 1,00 của 1,00

Given an array of non-negative integers. Each time, we can take the smallest integer out of the array, multiply it by 2, and push it back to the array.

Request: Implement function:

```
int leastAfter(vector<int>& nums, int k);
```

Where `nums` is the given array (the length of the array is between 1 and 100000). This function returns the smallest integer in the array after performing the operation `k` times (`k` is between 1 and 100000).

Example:

Given `nums = [2, 3, 5, 7]`.

In the 1st operation, we take 2 out and push back 4. The array is now `nums = [3, 4, 5, 7]`.

In the 2nd operation, we take 3 out and push back 6. The array is now `nums = [4, 5, 6, 7]`.

In the 3rd operation, we take 4 out and push back 8. The array is now `nums = [5, 6, 7, 8]`.

With `k = 3`, the result would be 5.

Note:

In this exercise, the libraries `iostream`, `string`, `cstring`, `climits`, `utility`, `vector`, `list`, `stack`, `queue`, `map`, `unordered_map`, `set`, `unordered_set`, `functional`, `algorithm` has been included and `namespace std` are used. You can write helper functions and classes. Importing other libraries is allowed, but not encouraged, and may result in unexpected errors.

For example:

Test	Result
<pre>vector<int> nums {2, 3, 5, 7}; int k = 3; cout << leastAfter(nums, k);</pre>	5

Answer: (penalty regime: 0, 0, 0, 5, 10, ... %)

Reset answer

```
1 int leastAfter(vector<int>& nums, int k) {
2     // STUDENT ANSWER
3     priority_queue<int, vector<int>, greater<int>> pq(nums.begin(), nums.end());
4     for(int i = 0; i < k; i++) {
5         int min = pq.top();
6         pq.pop();
7         pq.push(min*2);
8     }
9     return pq.top();
10 }
```

	Test	Expected	Got	
✓	<pre>vector<int> nums {2, 3, 5, 7}; int k = 3; cout << leastAfter(nums, k);</pre>	5	5	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

BÁCH KHOA E-LEARNING



WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

📍 268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

☎ (028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

✉ elearning@hcmut.edu.vn

Copyright 2007-2022 BKEL - Phát triển dựa trên Moodle