Đã bắt đầu vào lúc	Thứ năm, 12 Tháng mười 2023, 4:15 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ bảy, 21 Tháng mười 2023, 11:54 PM
Thời gian thực hiện	9 ngày 7 giờ
Điểm	7,00/7,00
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

[Eng] Given a queue of integers of even length, rearrange the elements by interleaving the first half of the queue with the second half of the queue.

Your task is to implement interleaveQueue function.

stack and queue are included.

[Vie] Cho 1 hàng đợi có số lượng phần tử là số chẵn, sắp xếp lại các phần tử theo quy tắc xen kẽ phần tử ở nửa đầu và nửa sau của hàng đợi.

Sinh viên cần hiện thực hàm interleaveQueue.

Thư viện stack và queue đã được thêm vào.

For example:

Test	Input	Result

queue<int> q;
int n; cin >> n;
for (int i = 0; i < n; i++){
int element; cin >> element;
q.push(element);
}
interleaveQueue(q);
while (!q.empty()){
cout << q.front() << ' ';
q.pop();
}
	4
	1 2 3 4
		1 3 2 4

queue<int> q;
int n; cin >> n;
for (int i = 0; i < n; i++){
int element; cin >> element;
q.push(element);
}
interleaveQueue(q);
while (!q.empty()){
cout << q.front() << ' ';
q.pop();
}
	6
	2 4 6 8 10 12
		2 8 4 10 6 12

Answer: (penalty regime: 0 %)

Reset answer

```
//Tạo 2 queue
void interleaveQueue(queue<int>& q){
    queue<int> odd;
    queue<int> even;
    unsigned int x, n = q.size();
    while(!q.empty()) {
        x = q.front();
        q.pop();
        if(q.size() >= n/2) odd.push(x);
        else even.push(x);
    }
    while(!odd.empty()) {
        q.push(odd.front());
        odd.pop();
        q.push(even.front());
        even.pop();
    }
}
```

	Test	Input	Expected	Got	

✔	queue<int> q;
	int n; cin >> n;
	for (int i = 0; i < n; i++){
	int element; cin >> element;
	q.push(element);
	}
	interleaveQueue(q);
	while (!q.empty()){
	cout << q.front() << ' ';
	q.pop();
	}
		4
		1 2 3 4
			1 3 2 4	1 3 2 4	✔

✔	queue<int> q;
	int n; cin >> n;
	for (int i = 0; i < n; i++){
	int element; cin >> element;
	q.push(element);
	}
	interleaveQueue(q);
	while (!q.empty()){
	cout << q.front() << ' ';
	q.pop();
	}
		6
		2 4 6 8 10 12
			2 8 4 10 6 12	2 8 4 10 6 12	✔

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

Research **queue** which is implemented in C library at http://www.cplusplus.com/reference/queue/queue/. You can use library **queue** in c++ for this question.

Using **queue**, complete function **bool isBipartite(vector<vector<int>> graph)** to determine if a graph is bipartite or not (the graph can be disconnected). In caat https://en.wikipedia.org/wiki/Bipartite_graph.

You can use below liberaries in this question.

```
#include <iostream>
#include <vector>
#include <queue>
```

For example:

Test	Result

`int G[6][6] = { {0, 1, 0, 0, 0, 1},`
`{1, 0, 1, 0, 0, 0},`
`{0, 1, 0, 1, 0, 0},`
`{0, 0, 1, 0, 1, 0},`
`{0, 0, 0, 1, 0, 1},`
`{1, 0, 0, 0, 1, 0} };`
`int n = 6;`
 	
`vector<vector<int>> graph(n, vector<int>());`
`for (int i = 0; i < n; ++i) {`
`for (int j = 0; j < n; ++j) {`
`if (G[i][j]) graph[i].push_back(j);`
`}`
`}`
 	
`isBipartite(graph) ? cout << "Yes" : cout << "No";`
	Yes

Answer: (penalty regime: 0 %)

Reset answer

```
bool isBipartite(vector<vector<int>> graph) {
    int n = graph.size();
    vector<int> colour(n);// 0: đ** có màu; 1: màu đỏ; -1: màu xanh
    queue<int> q;
    for(int i = 0; i < n; i++) {
        if(colour[i]) continue; // Nào có màu bỏ
        colour[i] = 1; //Tô màu đỉnh i màu đỏ
        // BFS
        // Ban đầu tô 1 đỉnh xong đưa nó vào queue
        for(q.push(i); !q.empty(); q.pop()) {
           int cur = q.front();
           for(int neighbor : graph[cur]) { // Chạy trong hàng, check những thằng nó nối
                if(!colour[neighbor]) { // Chưa có màu thì tô màu còn lại
                   colour[neighbor] = -colour[cur];
                   q.push(neighbor);
             }
                else if(colour[neighbor] == colour[cur]) return false; // 2 Thằng cạnh nhau cùng màu
        }
     }
```

```
20      }
21      return true;
22  }
```

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

Research **queue** which is implemented in C library at: http://www.cplusplus.com/reference/queue/queue/. You can use library **queue** in c++ for this question.

Using **queue**, complete function **void bfs(vector<vector<int>> graph, int start)** to traverse all the nodes of the graph from given start node using Breadth First Search algorithm and data structure **queue**, and print the order of visited nodes.

You can use below liberaries in this question.

```
#include <iostream>
#include <vector>
#include <queue>
```

For example:

Test	Result

`int init_graph[10][10] = {  {0, 1, 1, 0, 1, 0, 1, 0, 1, 0},`
`{0, 0, 1, 1, 0, 0, 0, 1, 0, 0},`
`{0, 1, 0, 0, 0, 1, 1, 0, 1, 1},`
`{1, 0, 0, 0, 0, 0, 0, 1, 0, 0},`
`{0, 1, 0, 0, 0, 0, 0, 1, 0, 0},`
`{1, 0, 1, 0, 1, 0, 0, 0, 1, 0},`
`{0, 0, 1, 1, 0, 1, 0, 0, 0, 0},`
`{1, 0, 0, 0, 0, 1, 1, 0, 1, 0},`
`{0, 0, 0, 0, 0, 1, 0, 1, 0, 1},`
`{1, 0, 1, 0, 1, 0, 0, 0, 1, 0} };`
`int n = 10;`
`vector<vector<int>> graph(n, vector<int>());`
`for (int i = 0; i < n; ++i) {`
`for (int j = 0; j < n; ++j) {`
`if (init_graph[i][j]) graph[i].push_back(j);`
`}`
`}`
 	
`bfs(graph, 0);`
	`0 1 2 4 6 8 3 7 5 9`

Answer: (penalty regime: 0 %)

Reset answer

```
1  void bfs(vector<vector<int>> graph, int start) {
2      int vertices = graph.size();
3      bool visited[vertices], first = true;
4      for(int i = 0; i < graph.size(); i++) {
5          visited[i] = false;
6      }
7      queue<int> q;
8      visited[start] = true;
9      q.push(start);
10     while(!q.empty()) {
11         start = q.front();
12         if(first) {
13             cout << start;
14             first = false;
15         }
16         else cout << " " << start;
17         q.pop();
18         for(int i = 0; i < graph[start].size(); i++) {
19             int t = graph[start][i];
20             if(!visited[t]) {
21                 visited[t] = true;
22                 q.push(t);
```

```
23                  }
24              }
25          }
26      }
```

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 all methods in class **Queue** with template type **T**. The description of each method is written as comment in frame code.

```
#ifndef QUEUE_H
#define QUEUE_H
#include "DLinkedList.h"
template<class T>
class Queue {
protected:
    DLinkedList<T> list;
public:
    Queue() {}
    void push(T item) ;
    T pop() ;
    T top() ;
    bool empty() ;
    int size() ;
    void clear() ;
};

#endif /* QUEUE_H */
```

You can use all methods in class **DLinkedList** without implementing them again. The description of class **DLinkedList** is written as comment in frame code.

```
template <class T>
class DLinkedList
{
public:
    class Node;     //forward declaration
protected:
    Node* head;
    Node* tail;
    int count;
public:
    DLinkedList() ;
    ~DLinkedList();
    void add(const T& e);
    void add(int index, const T& e);
    T removeAt(int index);
    bool removeItem(const T& removeItem);
    bool empty();
    int size();
    void clear();
    T get(int index);
    void set(int index, const T& e);
    int indexOf(const T& item);
    bool contains(const T& item);
};
```

For example:

Test	Result

`Queue<int> queue;`
`    assert(queue.empty());`
`    assert(queue.size() == 0);`	

Answer: (penalty regime: 0 %)

Reset answer

```
void push(T item) {
    // TODO: Push new element into the end of the queue
    this->list.add(item);
}

T pop() {
    // TODO: Remove an element in the head of the queue
    return this->list.removeAt(0);
}

T top() {
    // TODO: Get value of the element in the head of the queue
    return this->list.get(0);
}

bool empty() {
    // TODO: Determine if the queue is empty
    return this->list.empty();
}

int size() {
    // TODO: Get the size of the queue
    return this->list.size();
}

void clear() {
    // TODO: Clear all elements of the queue
    this->list.clear();
}
```

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

A nice number is a positive integer that contains only 2's and 5's.
Some nice numbers are: 2, 5, 22, 25, 52, 55, ...
Number 2 is the first nice number.
Given an integer N, return the Nth nice number.
Note: iostream, vector, queue are already included for you.

Constraint:
1 <= n <= 10^6

Example 1:
Input:
n = 5
Output:
52

Explanation:
The sequence of nice numbers is 2, 5, 22, 25, 52, 55, ...
The 5th number in this sequence is 52

Example 2:
Input:
n = 10000
Output:
2255522252225

For example:

Test	Input	Result

`int n;`
`cin >> n;`
`cout << nthNiceNumber(n) << endl;`
	5	52

`int n;`
`cin >> n;`
`cout << nthNiceNumber(n) << endl;`
	10000	2255522252225

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

Reset answer

```
// iostream, vector and queue are included
// You can write helper methods

long long nthNiceNumber(int n) {
    string nice = "";
    queue<string> q;
    q.push("2");
    q.push("5");
    while(n) {
        nice = q.front();
        q.pop();
        q.push(nice + "2");
        q.push(nice + "5");
        n--;
    }
```

```
16      return stol(nice);
17  }
```

	Test	Input	Expected	Got	
✔	int n; cin >> n; cout << nthNiceNumber(n) << endl;	5	52	52	✔
✔	int n; cin >> n; cout << nthNiceNumber(n) << endl;	10000	2255522252225	2255522252225	✔

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

Given a n*m grid where each cell in the grid can have a value of 0, 1 or 2, which has the following meaning:
1. Empty cell
2. This cell contains a fresh apple
3. This cell contains a rotten apple
After 1 second, the cell with rotten apple will rot all fresh apples in all the cells adjacent to it (i.e the cells (x+1, y), (x-1, y), (x, y+1), (x, y-1))
Determine the minimum time (in seconds) required to rot all apples. If this cannot be done, return -1.
Note: iostream, vector, and queue are already included.

Constraint:
1 <= n, m <= 500

Hint: Have you ever heard about breadth-first-search?

Example 1:
Input: grid = {{2,2,0,1}}
Output: -1
Explanation:
The grid is
2 2 0 1
The apple at (0, 3) cannot be rotten

Example 2:
Input: grid = {{0,1,2},{0,1,2},{2,1,1}}
Output: 1
Explanation:
The grid is
0 1 2
0 1 2
2 1 1
Apples at positions (0,2), (1,2), (2,0)
will rot apples at (0,1), (1,1), (2,2) and (2,1) after 1 second.

For example:

Test	Input	Result

`int rows, cols;`
`cin >> rows >> cols;`
`vector<vector<int>> grid(rows, vector<int>(cols));`
`for(int i = 0; i < rows; i++) {`
`  for(int j = 0; j < cols; j++) cin >> grid[i][j];`
`}`
`cout << secondsToBeRotten(grid);`
	1 4
	2 2 0 1
		-1

`int rows, cols;`
`cin >> rows >> cols;`
`vector<vector<int>> grid(rows, vector<int>(cols));`
`for(int i = 0; i < rows; i++) {`
`  for(int j = 0; j < cols; j++) cin >> grid[i][j];`
`}`
`cout << secondsToBeRotten(grid);`
	3 3
	0 1 2
	0 1 2
	2 1 1
		1

Answer: (penalty regime: 0 %)

Reset answer

```cpp
// iostream, vector and queue are included
// Hint: use breadth-first-search
#include <tuple>
int secondsToBeRotten(vector<vector<int>>& grid) {
    int n = grid.size(), m = grid[0].size();
    vector<vector<int>> visited = grid;
    queue<pair<int,int>> q; // chứa rotten apple
    int FOrange = 0;
    for(int i = 0; i < n; i++) { // Tìm rotten và fresh ban đầu
        for(int j = 0; j < m; j++) {
            if(visited[i][j] == 2) q.push({i,j});
            if(visited[i][j] == 1) FOrange++;
        }
    }

    if(FOrange == 0) return 0; // Hư trong 0 giây
    if(q.empty()) return -1; // không có rotten sao rot

    int second = -1;
    // 4 hướng
    vector<pair<int,int>> dir = {{1, 0},{-1, 0},{0, -1},{0, 1}};
    while(!q.empty()) {
        int size = q.size();
        while(size--){
            int x,y;
            std::tie(x,y) = q.front();
            q.pop();
            for(pair<int,int> d : dir) {
                int dx,dy;
                std::tie(dx,dy) = d;
                int i = x + dx;
                int j = y + dy;
                if(i >= 0 && i < n && j >= 0 && j < m && visited[i][j] == 1) {
                    visited[i][j] = 2;
                    FOrange--;
                    q.push({i,j});
                }
            }
        }
```

	Test	Input	Expected	Got	

✔	`int rows, cols;`
	`cin >> rows >> cols;`
	`vector<vector<int>> grid(rows, vector<int>(cols));`
	`for(int i = 0; i < rows; i++) {`
	`  for(int j = 0; j < cols; j++) cin >> grid[i][j];`
	`}`
	`cout << secondsToBeRotten(grid);`
		1 4
		2 2 0 1
			-1	-1	✔

✔	`int rows, cols;`
	`cin >> rows >> cols;`
	`vector<vector<int>> grid(rows, vector<int>(cols));`
	`for(int i = 0; i < rows; i++) {`
	`  for(int j = 0; j < cols; j++) cin >> grid[i][j];`
	`}`
	`cout << secondsToBeRotten(grid);`
		3 3
		0 1 2
		0 1 2
		2 1 1
			1	1	✔

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 integers.
Your task is to implement a function with following prototype:

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

The function returns the sum of the maximum value of every consecutive subarray of `nums` with fixed length `k`.

Note:
- The `iostream`, `vector`, `queue` and `deque` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions and classes.

For example:

Test	Result

`vector<int> nums {1, 2, 4, 3, 6};`
`int k = 3;`
`cout << sumOfMaxSubarray(nums, k);`
	14

Answer: (penalty regime: 0 %)

Reset answer

```
int sumOfMaxSubarray(vector<int>& nums, int k) {
    deque<int> q(k); // Mảng lưu index của giá trị lớn nhất mỗi mảng con
    int i;
    // Tìm phần tử lớn nhất trong mảng con đầu tiên (Sliding Window)
    for (i = 0; i < k; ++i) {
        while ((!q.empty()) && nums[i] >= nums[q.back()])  q.pop_back();
        q.push_back(i);
    }
    int sum = 0; // Tổng trả zề
    for (; i < nums.size(); ++i) {
        sum += nums[q.front()];
        // Xóa phần tử trong q, đã ngoài tầm của window
        while ((!q.empty()) && q.front() <= i - k) q.pop_front();
        // Nếu phần tử hiện tại lớn hơn cái trong kia, cái đó cút
        while ((!q.empty()) && nums[i] >= nums[q.back()]) q.pop_back();
        // Đưa index hiện tại vô cuối q
        q.push_back(i);
    }
    sum += nums[q.front()];
    return sum;
}
```

	Test	Expected	Got	

✔	`vector<int> nums {1, 2, 4, 3, 6};`
	`int k = 3;`
	`cout << sumOfMaxSubarray(nums, k);`
		14	14	✔

✔	`vector<int> nums {8016};`
	`int k = 1;`
	`cout << sumOfMaxSubarray(nums, k);`
		8016	8016	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.