

<b>Đã bắt đầu vào lúc</b>	Thứ năm, 12 Tháng mười 2023, 3:12 PM
<b>Tình trạng</b>	Đã hoàn thành
<b>Hoàn thành vào lúc</b>	Thứ sáu, 20 Tháng mười 2023, 9:10 PM
<b>Thời gian thực hiện</b>	8 ngày 5 giờ
<b>Điểm</b>	8,00/8,00
<b>Điểm</b>	<b>10,00</b> của 10,00 ( <b>100%</b> )

# Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

You are keeping score for a basketball game with some new rules. The game consists of several rounds, where the scores of past rounds may affect future rounds' scores.

At the beginning of the game, you start with an empty record. You are given a list of strings **ops**, where **ops[i]** is the operation you must apply to the record, with the following rules:

- A non-negative integer **x (from 0 to 9)** - record a new score of **x**
- '+' - Record a new score that is the sum of the previous two scores. It is guaranteed there will always be two previous scores.
- 'D' - Record a new score that is double the previous score. It is guaranteed there will always be a previous score.
- 'C' - Invalidate the previous score, removing it from the record. It is guaranteed there will always be a previous score.

Finally, return the sum of all scores in the record.

### For example:

ops = "52CD+"

- '5' - add to the record. Record now is [5]
- '2' - add to the record. Record now is [5,2]
- 'C' - invalid the previous score (2). Record now is [5]
- 'D' - Record new score that is double of previous score (5\*2). Record now is [5,10]
- '+' - Record a new score that is the sum of the previous two scores. Record now is [5,10,15]

Return the sum: 5+10+15 = 30

### For example:

Test	Result
cout << baseballScore("52CD+");	30
cout << baseballScore("524CD9++");	55

**Answer:** (penalty regime: 0 %)

Reset answer

```

1 //sử dụng hàm isdigit
2 int baseballScore(string ops){
3     /*TODO*/
4     stack<int> stack;
5     for(unsigned int i = 0; i < ops.size(); i++) {
6         if(isdigit(ops[i])) stack.push(ops[i] - '0');
7         if(ops[i] == 'D' && !stack.empty()) {
8             int num = stack.top();
9             num *= 2;
10            stack.push(num);
11        }
12        if(ops[i] == 'C' && !stack.empty()) {
13            stack.pop();
14        }
15        if(ops[i] == '+') {
16            int num1 = stack.top();
17            stack.pop();
18            int num2 = stack.top();
19            stack.push(num1);
20            stack.push(num1+num2);
21        }
22    }
23    int sum = 0;
24    while (!stack.empty()) {
25        sum += stack.top();
26        stack.pop();
27    }
28    return sum;
29 }
```

```
25     sum += stack.pop();
26     stack.pop();
27 }
28 return sum;
29 }
```

	Test	Expected	Got	
✓	cout << baseballScore("52CD+");	30	30	✓
✓	cout << baseballScore("524CD9++");	55	55	✓
✓	cout << baseballScore("5C4C2C11+D3");	11	11	✓

Passed all tests! ✓

Chỉnh xác

Điểm cho bài nộp này: 1,00/1,00.

## Câu hỏi 2

Chính xác

Điểm 1,00 của 1,00

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

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

#endif
```

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
Stack<int> stack; cout << stack.empty() << " " << stack.size();	1 0
Stack<int> stack;  int item[] = { 3, 1, 4, 5, 2, 8, 10, 12 }; for (int idx = 0; idx < 8; idx++) stack.push(item[idx]);  assert(stack.top() == 12);  stack.pop(); stack.pop();  cout << stack.top();	8

**Answer:** (penalty regime: 0 %)

Reset answer

```

1 void push(T item) {
2     // TODO: Push new element into the top of the stack
3     this->list.add(item);
4 }
5
6 T pop() {
7     // TODO: Remove an element on top of the stack
8     return this->list.removeAt(this->list.size() - 1);
9 }
10
11 T top() {
12     // TODO: Get value of the element on top of the stack
13     return this->list.get(this->list.size() - 1);
14 }
15
16 bool empty() {
17     // TODO: Determine if the stack is empty
18     return this->list.empty();
19 }
20
21 int size() {
22     // TODO: Get the size of the stack
23     return this->list.size();
24 }
25
26 void clear() {
27     // TODO: Clear all elements of the stack
28     this->list.clear();
29 }

```

	Test	Expected	Got	
✓	Stack<int> stack; cout << stack.empty() << " " << stack.size();	1 0	1 0	✓
✓	Stack<int> stack;  int item[] = { 3, 1, 4, 5, 2, 8, 10, 12 }; for (int idx = 0; idx < 8; idx++) stack.push(item[idx]);  assert(stack.top() == 12);  stack.pop(); stack.pop();  cout << stack.top();	8	8	✓

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 `nums[]` of size `N` having distinct elements, the task is to find the next greater element for each element of the array. Next greater element of an element in the array is the nearest element on the right which is greater than the current element. If there does not exist a next greater of a element, the next greater element for it is `-1`.

Note: `iostream`, `stack` and `vector` are already included

Constraints:

$1 \leq \text{nums.length} \leq 10^5$

$0 \leq \text{nums}[i] \leq 10^9$

Example 1:

Input:

`nums = {15, 2, 4, 10}`

Output:

`{-1, 4, 10, -1}`

Example 2:

Input:

`nums = {1, 4, 6, 9, 6}`

Output:

`{4, 6, 9, -1, -1}`

For example:

Test	Input	Result
<pre>int N; cin &gt;&gt; N; vector&lt;int&gt; nums(N); for(int i = 0; i &lt; N; i++) cin &gt;&gt; nums[i]; vector&lt;int&gt; greaterNums = nextGreater(nums); for(int i : greaterNums)     cout &lt;&lt; i &lt;&lt; ' '; cout &lt;&lt; '\n';</pre>	<pre>4 15 2 4 10</pre>	<pre>-1 4 10 -1</pre>
<pre>int N; cin &gt;&gt; N; vector&lt;int&gt; nums(N); for(int i = 0; i &lt; N; i++) cin &gt;&gt; nums[i]; vector&lt;int&gt; greaterNums = nextGreater(nums); for(int i : greaterNums)     cout &lt;&lt; i &lt;&lt; ' '; cout &lt;&lt; '\n';</pre>	<pre>5 1 4 6 9 6</pre>	<pre>4 6 9 -1 -1</pre>

**Answer:** (penalty regime: 0 %)

Reset answer

```
1 // iostream, stack and vector are included
2 vector<int> nextGreater(vector<int>& arr){
3     for(int i = 0; i < arr.size() - 1; i++) {
4         for(int j = i+1; j < arr.size(); j++) {
5             if(arr[j] >= arr[i]) {
6                 arr[i] = arr[j];
7                 break;
8             }
9             if(j == arr.size() - 1) arr[i] = -1;
```

```
10     }
11     }
12     arr[arr.size() - 1] = -1;
13     return arr;
14 }
```

	Test	Input	Expected	Got	
✓	<pre>int N; cin &gt;&gt; N; vector&lt;int&gt; nums(N); for(int i = 0; i &lt; N; i++) cin &gt;&gt; nums[i]; vector&lt;int&gt; greaterNums = nextGreater(nums); for(int i : greaterNums)     cout &lt;&lt; i &lt;&lt; ' '; cout &lt;&lt; '\n';</pre>	<pre>4 15 2 4 10</pre>	<pre>-1 4 10 -1</pre>	<pre>-1 4 10 -1</pre>	✓
✓	<pre>int N; cin &gt;&gt; N; vector&lt;int&gt; nums(N); for(int i = 0; i &lt; N; i++) cin &gt;&gt; nums[i]; vector&lt;int&gt; greaterNums = nextGreater(nums); for(int i : greaterNums)     cout &lt;&lt; i &lt;&lt; ' '; cout &lt;&lt; '\n';</pre>	<pre>5 1 4 6 9 6</pre>	<pre>4 6 9 -1 -1</pre>	<pre>4 6 9 -1 -1</pre>	✓

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

Given string **S** representing a **postfix expression**, the task is to evaluate the expression and find the final value. Operators will only include the basic arithmetic operators like **\***, **/**, **+** and **-**.

**Postfix expression:** The expression of the form "a b operator" (ab+) i.e., when a pair of operands is followed by an operator.

**For example:** Given string S is "2 3 1 \* + 9 -". If the expression is converted into an infix expression, it will be  $2 + (3 * 1) - 9 = 5 - 9 = -4$ .

**Requirement:** Write the function to evaluate the value of postfix expression.

**For example:**

Test	Result
cout << evaluatePostfix("2 3 1 * + 9 -");	-4
cout << evaluatePostfix("100 200 + 2 / 5 * 7 +");	757

**Answer:** (penalty regime: 0 %)

Reset answer

```

1  int evaluatePostfix(string expr){
2      /*TODO*/
3      stack<string> Postfix;
4      int b = 0, c = 0;
5      string a = "";
6      unsigned int i = 0;
7      while(i != expr.size()) {
8          if(expr[i] == ' ') i++;
9          else if(expr[i] >= '0' && expr[i] <= '9') {
10             while(expr[i] != ' ') {
11                 a += expr[i];
12                 i++;
13             }
14             Postfix.push(a);
15             a = "";
16             i++;
17         }
18         else {
19             b = stoi(Postfix.top());
20             Postfix.pop();
21             c = stoi(Postfix.top());
22             Postfix.pop();
23             switch(expr[i]) {
24                 case '+': {
25                     b = b + c;
26                     break;
27                 }
28                 case '-': {
29                     b = c - b;
30                     break;
31                 }
32                 case '*': {
33                     b = b * c;
34                     break;
35                 }
36                 case '/': {
37                     b = c / b;

```

	Test	Expected	Got	
✓	cout << evaluatePostfix("2 3 1 * + 9 -");	-4	-4	✓
✓	cout << evaluatePostfix("100 200 + 2 / 5 * 7 +");	757	757	✓

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 Maze is given as 5\*5 binary matrix of blocks and there is a rat initially at the upper left most block i.e., maze[0][0] and the rat wants to eat food which is present at some given block in the maze (fx, fy). In a maze matrix, 0 means that the block is a dead end and 1 means that the block can be used in the path from source to destination. The rat can move in any direction (not diagonally) to any block provided the block is not a dead end.

Your task is to implement a function with following prototype to check if there exists any path so that the rat can reach the food or not:

```
bool canEatFood(int maze[5][5], int fx, int fy);
```

Template:

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <stack>
#include <vector>
using namespace std;

class node {
public:
    int x, y;
    int dir;
    node(int i, int j)
    {
        x = i;
        y = j;

        // Initially direction
        // set to 0
        dir = 0;
    }
};
```

Some suggestions:

- X : x coordinate of the node
- Y : y coordinate of the node
- dir : This variable will be used to tell which all directions we have tried and which to choose next. We will try all the directions in anti-clockwise manner starting from up.
  - If dir=0 try up direction.
  - If dir=1 try left direction.
  - If dir=2 try down direction.
  - If dir=3 try right direction.

For example:

Test	Result
<pre>// Maze matrix int maze[5][5] = {     { 1, 0, 1, 1, 0 },     { 1, 1, 1, 0, 1 },     { 0, 1, 0, 1, 1 },     { 1, 1, 1, 1, 0 },     { 1, 0, 0, 1, 0 } };  // Food coordinates int fx = 1, fy = 4;  cout &lt;&lt; canEatFood(maze, fx, fy);</pre>	1
<pre>// Maze matrix int maze[5][5] = {     { 1, 0, 1, 1, 0 },     { 1, 1, 1, 0, 0 },     { 0, 1, 0, 1, 1 },     { 0, 1, 0, 1, 0 },     { 0, 1, 1, 1, 0 } };  // Food coordinates int fx = 2, fy = 3;  cout &lt;&lt; canEatFood(maze, fx, fy);</pre>	1

**Answer:** (penalty regime: 0 %)

Reset answer

```
1 bool canEatFood(int maze[5][5], int fx, int fy){
2     /*TODO*/
3     int x = 0, y = 0;
4     stack<node> Path;
5     node cur(x,y);
6     bool saveStep[5][5];
7     memset(saveStep, true, sizeof(saveStep));
8     Path.push(cur); // Khởi tạo Stack và vị trí đầu tiên
9     while(!Path.empty()) {
10         cur = Path.top();
11         int direction = cur.dir;
12         x = cur.x; y = cur.y;
13         cur.dir++;
14         Path.pop();
15         Path.push(cur);
16         if(x == fx && y == fy) return true; // Đụng thức ăn
17         switch(direction) {
18             case 0: {
19                 if(x-1 >= 0 && saveStep[x-1][y] && maze[x-1][y]) {
20                     node cur1(x-1,y);
21                     saveStep[x-1][y] = false;
22                     Path.push(cur1);
23                 }
24                 break;
25             }
26             case 1: {
27                 if(y-1 >=0 && saveStep[x][y-1] && maze[x][y-1]) {
28                     node cur1(x,y-1);
29                     saveStep[x][y-1] = false;
30                     Path.push(cur1);
31                 }
32                 break;
33             }
34             case 2: {
35                 if(x+1 < 5 && saveStep[x+1][y] && maze[x+1][y]) {
36                     node cur1(x+1, y);
```

```

37         saveStep[x+1][y] = false;
38         Path.push(cur1);
39     }
40     break;
41 }
42 case 3: {
43     if(y+1 < 5 && saveStep[x][y+1] && maze[x][y+1]) {
44         node cur1(x,y+1);
45         saveStep[x][y+1] = false;
46         Path.push(cur1);
47     }
48     break;
49 }
50 default: {
51     saveStep[cur.x][cur.y] = true;
52     Path.pop();
53 }
54 }
55 }
56 return false; // Stack rỗng = không có đường đi
57 }

```

	Test	Expected	Got	
✓	<pre> // Maze matrix int maze[5][5] = {     { 1, 0, 1, 1, 0 },     { 1, 1, 1, 0, 1 },     { 0, 1, 0, 1, 1 },     { 1, 1, 1, 1, 0 },     { 1, 0, 0, 1, 0 } };  // Food coordinates int fx = 1, fy = 4;  cout &lt;&lt; canEatFood(maze, fx, fy); </pre>	1	1	✓
✓	<pre> // Maze matrix int maze[5][5] = {     { 1, 0, 1, 1, 0 },     { 1, 1, 1, 0, 0 },     { 0, 1, 0, 1, 1 },     { 0, 1, 0, 1, 0 },     { 0, 1, 1, 1, 0 } };  // Food coordinates int fx = 2, fy = 3;  cout &lt;&lt; canEatFood(maze, fx, fy); </pre>	1	1	✓

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 string **S** of characters, a *duplicate removal* consists of choosing two adjacent and equal letters, and removing them.

We repeatedly make duplicate removals on **S** until we no longer can.

Return the final string after all such duplicate removals have been made.

Included libraries: vector, list, stack

**For example:**

Test	Result
cout << removeDuplicates("abbaca");	ca
cout << removeDuplicates("aab");	b

**Answer:** (penalty regime: 0 %)

Reset answer

```

1 string reverse(string S) {
2     for(unsigned int i = 0; i < S.length()/2; i++) {
3         swap(S[i], S[S.length() - i - 1]);
4     }
5     return S;
6 }
7 string removeDuplicates(string S){
8     /*TODO*/
9     stack<char> checkDup;
10    unsigned int i = 0;
11    while(i != S.size()) {
12        if(!checkDup.empty() && checkDup.top() == S[i]) checkDup.pop();
13        else checkDup.push(S[i]);
14        i++;
15    }
16    S = "";
17    while(!checkDup.empty()) {
18        S += checkDup.top();
19        checkDup.pop();
20    }
21    return reverse(S);
22 }
```

	Test	Expected	Got	
✓	cout << removeDuplicates("abbaca");	ca	ca	✓
✓	cout << removeDuplicates("aab");	b	b	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

## Câu hỏi 7

Chính xác

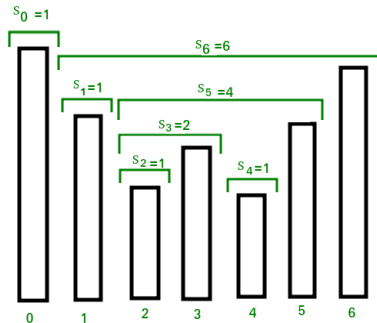
Điểm 1,00 của 1,00

### Vietnamese version:

Bài toán stock span là một bài toán về chủ đề kinh tế tài chính, trong đó ta có thông tin về giá của một cổ phiếu qua từng ngày. Mục tiêu của bài toán là tính *span* của giá cổ phiếu ở từng ngày.

*Span* của giá cổ phiếu tại ngày thứ  $i$  (ký hiệu là  $S_i$ ) được định nghĩa là *số ngày liên tục nhiều nhất liền trước ngày thứ  $i$  có giá cổ phiếu thấp hơn, cộng cho 1 (cho chính nó)*.

Ví dụ, với chuỗi giá cổ phiếu là [100, 80, 60, 70, 60, 75, 85].



1. Ngày thứ 0 không có ngày liền trước nên  $S_0$  bằng 1.
2. Ngày thứ 1 có giá nhỏ hơn giá ngày thứ 0 nên  $S_1$  bằng 1.
3. Ngày thứ 2 có giá nhỏ hơn giá ngày thứ 1 nên  $S_2$  bằng 1.
4. Ngày thứ 3 có giá **lớn hơn** giá ngày thứ 2 nên  $S_3$  bằng 2.
5. Ngày thứ 4 có giá nhỏ hơn giá ngày thứ 3 nên  $S_4$  bằng 1.
6. Ngày thứ 5 có giá **lớn hơn giá ngày thứ 4, 3, 2** nên  $S_5$  bằng 4.
7. Ngày thứ 6 có giá **lớn hơn giá ngày thứ 5, 4, 3, 2, 1** nên  $S_6$  bằng 6.

Kết quả sẽ là [1, 1, 1, 2, 1, 4, 6].

**Yêu cầu.** Viết chương trình tính toán chuỗi span từ chuỗi giá cổ phiếu từ đầu vào.

**Input.** Các giá trị giá cổ phiếu, cách nhau bởi các ký tự khoảng trắng, được đưa vào standard input.

**Output.** Các giá trị span, cách nhau bởi một khoảng cách, được xuất ra standard output.

(Nguồn: Geeks For Geeks)

=====

### Phiên bản tiếng Anh:

The stock span problem is a financial problem where we have a series of daily price quotes for a stock and we need to calculate the span of the stock's price for each day.

The span  $S_i$  of the stock's price on a given day  $i$  is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than its price on the given day, plus 1 (for itself).

For example: take the stock's price sequence [100, 80, 60, 70, 60, 75, 85]. (See image above)

The given input span for 100 will be 1, 80 is smaller than 100 so the span is 1, 60 is smaller than 80 so the span is 1, 70 is greater than 60 so the span is 2 and so on.

Hence the output will be [1, 1, 1, 2, 1, 4, 6].



**Requirement.** Write a program to calculate the spans from the stock's prices.

**Input.** A list of whitespace-delimited stock's prices read from standard input.

**Output.** A list of space-delimited span values printed to standard output.

(Source: Geeks For Geeks)

**For example:**

Input	Result
100 80 60 70 60 75 85	1 1 1 2 1 4 6
10 4 5 90 120 80	1 1 2 4 5 1

**Answer:** (penalty regime: 0 %)

Reset answer

```
1 vector<int> stock_span(const vector<int>& ns) {
2     vector<int> output;
3     stack<int> stockSpan;
4     int n = ns.size();
5
6     for (int i = 0; i < n; i++) {
7         while (!stockSpan.empty() && ns[stockSpan.top()] < ns[i]) {
8             stockSpan.pop();
9         }
10        if (stockSpan.empty()) {
11            output.push_back(i + 1);
12        } else {
13            output.push_back(i - stockSpan.top());
14        }
15        stockSpan.push(i);
16    }
17    return output;
18 }
```



	Input	Expected	Got	
✓	100 80 60 70 60 75 85	1 1 1 2 1 4 6	1 1 1 2 1 4 6	✓
✓	10 4 5 90 120 80	1 1 2 4 5 1	1 1 2 4 5 1	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 8

Chính xác

Điểm 1,00 của 1,00

Given a string **s** containing just the characters '(', ')', '[', ']', '{', and '}'. Check if the input string is valid based on following rules:

- 1. Open brackets must be closed by the same type of brackets.
- 2. Open brackets must be closed in the correct order.

For example:

- String "[()]" is a valid string, also "[()]".
- String "[()]" is **not** a valid string.

Your task is to implement the function

```
bool isValidParentheses (string s){
    /*TODO*/
}
```

Note: The library stack of C++ is included.

For example:

Test	Result
cout << isValidParentheses("[");	1
cout << isValidParentheses("[]()");	1
cout << isValidParentheses("[]");	0

Answer: (penalty regime: 0 %)

Reset answer

```
1 bool isValidParentheses (string s){
2     /*TODO*/
3     stack<char> BracketPars;
4     unsigned int i = 0;
5     while(i != s.size()) {
6         switch(s[i]) {
7             case '(': {
8                 BracketPars.push(s[i]);
9                 break;
10            }
11            case '[': {
12                BracketPars.push(s[i]);
13                break;
14            }
15            case '{': {
16                BracketPars.push(s[i]);
17                break;
18            }
19            case ')': {
20                if(BracketPars.empty()) return false;
21                char check = BracketPars.top();
22                if(check == '(') {
23                    BracketPars.pop();
24                    break;
25                }
26                else return false;
27            }
28            case ']': {
29                if(BracketPars.empty()) return false;
30                char check = BracketPars.top();
31                if(check == '[') {
32                    BracketPars.pop();
33                    break;
34                }
35                else return false;
36            }
37            case '}': {
38                if(BracketPars.empty()) return false;
39                char check = BracketPars.top();
40                if(check == '{') {
41                    BracketPars.pop();
42                    break;
43                }
44                else return false;
45            }
46        }
47        i++;
48    }
49    return BracketPars.empty();
50 }
```

```

34         }
35         else return false;
36     }
37     case '}' : {
38         if(BracketPars.empty()) return false;
39         char check = BracketPars.top();
40         if(check == '{') {
41             BracketPars.pop();
42             break;
43         }
44         else return false;
45     }
46 }
47 i++;
48 }
49 if(BracketPars.empty()) return true;
50 else return false;
51 }

```

	Test	Expected	Got	
✓	cout << isValidParentheses("[]()");	1	1	✓
✓	cout << isValidParentheses("[ ]");	0	0	✓

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)

Copyright 2007-2022 BKEI - Phát triển dựa trên Moodle