Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Tư, 18 tháng 9 2024, 7:32 AM
Kết thúc lúc	Thứ Ba, 24 tháng 9 2024, 4:25 PM
Thời gian thực hiện	6 Các ngày 8 giờ
Điểm	10,80/12,00
Điểm	**9,00** trên 10,00 (**90**%)

Câu hỏi 1

Đúng

Đạt điểm 1,00 trên 1,00

Implement function

```
void printArray(int n){}
```

to print 0, 1, 2, ..., n (n is positive integer and has no space at the end).

Please note that you can't using key work for, while, goto (even in variable names, comment).

For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`printArray(5);`	`0, 1, 2, 3, 4, 5`
`printArray(10);`	`0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10`

Answer: (penalty regime: 0, 0, 0, 5, 10, 15, ... %)

Reset answer

```
void printArray(int n)
{
    //base case
    if(n == 0){
        cout << 0;
    }
    //recursive case
    else{
        printArray(n-1);
        cout << ", " << n;
    }
}
```

	Test	Expected	Got	
✓	`printArray(5);`	`0, 1, 2, 3, 4, 5`	`0, 1, 2, 3, 4, 5`	✓
✓	`printArray(10);`	`0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10`	`0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **2**

Đúng

Đạt điểm 1,00 trên 1,00

Given a positive number, print following a pattern without using any loop.

```
Input: n = 16
Output: 16, 11, 6, 1, -4, 1, 6, 11, 16 (has no space at the end)

Input: n = 10
Output: 10, 5, 0, 5, 10 (has no space at the end)
```

We basically first reduce 5 one by one until we reach a negative or 0. After we reach 0 or negative, we one add 5 until we reach n.

Note: Please note that you can't using key work for, while, goto (even in variable names, comment).

You can implement other recursive functions if needed.

For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`printPattern(14);`	`14 9 4 -1 4 9 14`

Answer: (penalty regime: 0 %)

Reset answer

```
1  void printPattern(int n)
2  {
3      //base case
4      if(n <= 0){
5          cout << n;
6      }
7      //recursive case
8      else{
9          cout << n << " ";
10         printPattern(n-5);
11         cout << " " << n;
12     }
13 }
```

	Test	Expected	Got	
✓	`printPattern(14);`	`14 9 4 -1 4 9 14`	`14 9 4 -1 4 9 14`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **3**

Đúng

Đạt điểm 1,00 trên 1,00

Implement function

```
int findMax(int* arr, int length){}
```

to find the largest element using recursion (with length is the number of elements in integer array arr).

Please note that you can't using key work for, while, goto (even in variable names, comment).

For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`int arr[] = {10, 5, 7, 9, 15, 6, 11, 8, 12, 2};` `cout << findMax(arr, 10);`	15

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

Reset answer

```
1  int findMax(int *arr, int length) {
2      //base case
3      if(length == 1){
4          return arr[0];
5      }
6      //recursive case
7      else{
8          return max(arr[length-1], findMax(arr, length-1));
9      }
10 }
```

	Test	Expected	Got	
✓	`int arr[] = {10, 5, 7, 9, 15, 6, 11, 8, 12, 2};` `cout << findMax(arr, 10);`	15	15	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **4**

Đúng một phần

Đạt điểm 0,80 trên 1,00

Implement function

```
bool isPalindrome(string str){}
```

to check if the given non empty string is palindrome, else not palindrome using recursion.

In test case, for extra point, we will have some palindrome sentences (All remaining test cases are words).

Please note that you can't using key work for, while, goto (even in variable names, comment).

For this exercise, we have #include <iostream>, #include <string.h> and using namespace std;

For example:

Test	Result
`cout << isPalindrome("mom");`	1
`cout << isPalindrome("do geese see god");`	1

Answer: (penalty regime: 0 %)

Reset answer

```
bool isPalindrome(string str) {
    //base case
    if(str.length() <= 1){
        return true;
    }
    //recursive case
    else{
        if(str[0] == str[str.length()-1]){
            return isPalindrome(str.substr(1, str.length()-2));
        }
        else{
            return false;
        }
    }
}
```

	Test	Expected	Got	
✓	`cout << isPalindrome("mom");`	1	1	✓
✗	`cout << isPalindrome("do geese see god");`	1	0	✗

Some hidden test cases failed, too.

Show differences

Đúng một phần

Marks for this submission: 0,80/1,00.

Câu hỏi **5**

Đúng

Đạt điểm 1,00 trên 1,00

Give two positive integers a and b, implement function

```
int findGCD(int a, int b){}
```

to find **GCD** (Greatest Common Divisor) of a and b using recursion.

Please note that you can't using key work for, while, goto (even in variable names, comment).

For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`cout << findGCD(124,32);`	4

Answer: (penalty regime: 0 %)

Reset answer

```
int findGCD(int a, int b)
{
    //base case
    if(b == 0){
        return a;
    }
    //recursive case
    else{
        return findGCD(b, a%b);
    }
}
```

	Test	Expected	Got	
✓	`cout << findGCD(124,32);`	4	4	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **6**

Sai

Đạt điểm 0,00 trên 1,00

String s contains lowercase letters, digits, "(" and ")", satisfying the following rules:

- Two digits cannot be adjacent.
- Two "(" cannot be adjacent.
- One "(" and one ")" cannot be adjacent.
- After any digit, there must be "(".
- The quantities of "(" and ")" are equal.

Change string s until new string t created, t contains only lowercase letters. These are changing rules:

- Sub-strings with form "n(p)", can change to "pp...p" (n times p), where n is a digit and p is a string.
- If p still contains "(", ")" or digits, continue to implement the above changing method.

Request: Implement function

```
expand(string s);
```

Where `s` is a string with the above form; return the result is a string containing only lowercase letters.

Example:

- String "2(ab3(cde)x)" changes into "abcdecdecdexabcdecdecdex".
- String "2(x0👍)3(z)" changes into "xxzzz".

Note: In this exercise, libraries `iostream`, `string` and `using namespace std;` have been used. You can add other functions for your answer, but you are not allowed to add other libraries.

For example:

Test	Result
`cout << expand("2(ab3(cde)x)") << "\n";`	`abcdecdecdexabcdecdecdex`
`cout << expand("2(x0(y))3(z)") << "\n";`	`xxzzz`

Answer: (penalty regime: 0 %)

Reset answer

```
string expand(string s)
{
    //base case
    if(s.length() == 0){
        return "";
    }
    //recursive case
    else{
        int i = 0;
        while(i < s.length() && s[i] != '('){
            i++;
        }
        if(i == s.length()){
            return s;
        }
        else{
            int j = i+1;
            int count = 1;
            while(j < s.length() && count != 0){
                if(s[j] == '('){
```

```
                    count++;
                }
                else if(s[j] == ')'){
                    count--;
                }
                j++;
            }
            string temp = s.substr(i+1, j-i-2);
            string result = "";
            for(int k = 0; k < s[i-1]-'0'; k++){
                result += expand(temp);
            }
            return s.substr(0, i-1) + result + expand(s.substr(j));
        }
    }
}
```

Syntax Error(s)

```
__tester__.cpp: In function 'std::string expand(std::string)':
__tester__.cpp:16:17: error: comparison of integer expressions of different signedness: 'int' and
'std::__cxx11::basic_string<char>::size_type' {aka 'long unsigned int'} [-Werror=sign-compare]
   16 |         while(i < s.length() && s[i] != '('){
      |               ~~^~~~~~~~~~~~
__tester__.cpp:19:14: error: comparison of integer expressions of different signedness: 'int' and
'std::__cxx11::basic_string<char>::size_type' {aka 'long unsigned int'} [-Werror=sign-compare]
   19 |         if(i == s.length()){
      |            ~~^~~~~~~~~~~~~
__tester__.cpp:25:21: error: comparison of integer expressions of different signedness: 'int' and
'std::__cxx11::basic_string<char>::size_type' {aka 'long unsigned int'} [-Werror=sign-compare]
   25 |             while(j < s.length() && count != 0){
      |                   ~~^~~~~~~~~~~~
cc1plus: all warnings being treated as errors
```

Sai

Marks for this submission: 0,00/1,00.

Câu hỏi **7**

Đúng

Đạt điểm 1,00 trên 1,00

Give a positive integer x, implement recursive function

```
void printHailstone(int number){}
```

to print the Hailstone Sequence of a given number upto 1 (no space at the end).

Hailstone Sequences follow these rules:

- If a number is even, divide it by 2
- If a number is odd, multiply it by 3 and add 1.

Example:

```
If number = 5. 5 is odd number so next number is 5*3 + 1 = 16. 16 is even number so next number is 16/2
= 8...
Finally, we get Hailstone sequence: 5 16 8 4 2 1.
```

You can find more information at: https://diendantoanhoc.net/topic/89145-d%C3%A3y-s%E1%BB%91-hailstone/

Note: Please note that you can't using key work for, while, goto (even in variable names, comment).

You can implement other recursive functions if needed.

For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`printHailstone(32);`	`32 16 8 4 2 1`

Answer: (penalty regime: 0 %)

Reset answer

```
void printHailstone(int number)
{
    /*
     * STUDENT ANSWER
     */
    //base case
    if(number == 1){
        cout << 1;
    }
    //recursive case
    else{
        cout << number << " ";
        if(number % 2 == 0){
            printHailstone(number/2);
        }
        else{
            printHailstone(number*3 + 1);
        }
    }
}
```

	Test	Expected	Got	
✓	`printHailstone(32);`	`32 16 8 4 2 1`	`32 16 8 4 2 1`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **8**

Đúng

Đạt điểm 1,00 trên 1,00

Function

```
int myArrayToInt(char* str, int n){}
```

takes a **string str** (which represents an positive decimal number), **n** is the number of elements in the string as arguments and returns its value.

Please note that you can't using key work for, while, goto (even in variable names, comment)

For this exercise, we have #include <iostream>, #include <string.h> and using namespace std;

For example:

Test	Result
char str[] = "2020"; printf("%d", myArrayToInt(str, 4));	2020

Answer: (penalty regime: 0 %)

Reset answer

```
int myArrayToInt(char *str, int n)
{
    /*
     * STUDENT ANSWER
     */
    //base case
    if(n == 1){
        return str[0] - '0';
    }
    //recursive case
    else{
        return (str[n-1] - '0') + myArrayToInt(str, n-1) * 10;
    }
}
```

	Test	Expected	Got	
✓	`char str[] = "2020";` `printf("%d", myArrayToInt(str, 4));`	2020	2020	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi 9

Đúng

Đạt điểm 1,00 trên 1,00

Give two positive integers a and b, implement function

```
int findLCM(int a, int b){}
```

to find **LCM** (Lowest Common Multiple) of a and b using recursion.

Please note that you can't using key work for, while, goto (even in variable names, comment).

For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
cout << findLCM(10, 102);	510

Answer: (penalty regime: 0 %)

Reset answer

```
// Function to find GCD using recursion
int findGCD(int a, int b)
{
    // Base case
    if (b == 0)
    {
        return a;
    }
    // Recursive case
    return findGCD(b, a % b);
}

// Function to find LCM using recursion
int findLCM(int a, int b)
{
    // Base case
    if (a == 0 || b == 0)
    {
        return 0;
    }
    // Recursive case
    return (a * b) / findGCD(a, b);
}
```

	Test	Expected	Got	
✓	`cout << findLCM(10, 102);`	`510`	`510`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **10**

Đúng

Đạt điểm 1,00 trên 1,00

Given a string `s` consisting only of `'('` and `')'`.
Your task is to implement a function with following prototype:

```
int mininumBracketAdd(string s);
```

The function returns the mininum number of brackets needed to be inserted to `s` so that the brackets are balanced.

More info:
A sequence of brackets is balanced when there are no unmatched brackets.
Example: `()(())` is balanced, but `))()` is not.

Note:
- The `iostream` library has been used and `namespace std` is being used. No other libraries are allowed.
- Using loop keywords (`for`, `while`, `do`) are not allowed, even in comments and variable names.
- You can write helper functions.

For example:

Test	Result
`cout << mininumBracketAdd("))((");`	5

Answer: (penalty regime: 0 %)

Reset answer

```
// Helper function to recursively calculate the minimum number of brackets needed
int helper(const string &s, int index, int balance)
{
    // Base case: if we have reached the end of the string
    if (index == static_cast<int>(s.length()))
    {
        return abs(balance);
    }

    // Recursive case
    if (s[index] == '(')
    {
        // Increment balance f an opening parenthesis
        return helper(s, index + 1, balance + 1);
    }
    else
    {
        // Decrement balance f a closing parenthesis
        if (balance > 0)
        {
            return helper(s, index + 1, balance - 1);
        }
        else
        {
            // If balance is zero or negative, we need to add an opening parenthesis
            return 1 + helper(s, index + 1, balance);
        }
    }
}

// Function to find the minimum number of brackets needed to balance the string
int mininumBracketAdd(string s)
{
    return helper(s, 0, 0);
}
```

	Test	Expected	Got	
✓	`cout << mininumBracketAdd(")))((");`	5	5	✓
✓	`cout << mininumBracketAdd("))()))()()");`	4	4	✓
✓	`cout << mininumBracketAdd("");`	0	0	✓
✓	`cout << mininumBracketAdd(")()))()))())()))))(");`	12	12	✓
✓	`cout << mininumBracketAdd(")())(((()()())()()))()()))(()(((((()(()()()))()((()(()))))))` `()(((()(((()(((()()(())(()))))))(()))");`	10	10	✓

✓	`cout << mininumBracketAdd(")(()())()))))(((())))))()))))()))))))(()))(((()((((()((()(()))`
	`())(()(()((((((()))(((()))))(()))((()()))(())(()((((()())()()())(())()))(()))()))(((((()`
	`(()))())))()()()(()))()))()())()(()()))())())()))())())(((()()(())((()()((((()(((`
	`(((((((()))()()())(()()()))()()()())()(()(())())(()))(((((()((((((()((()))))))()))((()()))))`
	`()()((((()()()(()())(((((()((((((()))(()))))(()))))()())())((()))()())()((()()))(())((((()`
	`((()))((((()()(()((((()((()(()))()))))))()(()(())()())((((()(()(())()())()(()()))((()))()`
	`())()))))((()())(())()))(()))()))(())))()))())()()(()()((((((()(()()(()()))(()(()(())()((((()`
	`((()(()(()))))()())(((()()(((()))(()))()(())()((()()(((((()))((())(()())()))))()(()))((()))`
	`(()())()))()())()))(()((()()()((()()))(())(())((()))()())(())()((())((()(())(()))(())()`
	`((()))((()())((()))((((((()))()()()(()(()))(()()()()))))()))))()()))(())(((((((((()))`
	`(()))))(()))(()))(()((()()((()))))(()(())((((()))(()))))(()((()((()(((()))()))(()((())`
	`()))))(()))(()()))()(((((()(((()()(()))()((()(((()");`
		70	70	✓

	Test	Expected	Got	
✓	cout << mininumBracketAdd(")()))))))()())(((((()((()()))()((()()((())((()))()))())())()))() (()()))((()()((((()))()))()((()))()))))()())(()((((((()))()))(()))()))())))))(()))((()(())))) ((()(())()()((())(())()()))((((((((())()(())()()((((())(())(()()))))((())()())()(())) ()))((((()())((())((())(((()()()(())()()(()))(((())))()((()(())))))(()((())()()())()))(((() (((()()()(()())(()(())(()(())()())((((()()))))(())(()(()((((())()((()(())))()))((()() (())((()))))(()))))()()(()))()(((()()((((())(()(()()()()()((()))(((((()(()()()((()()))() (())())(())))))(()(())(()(()()((((((((((()((()((())((()(())((()((()))()())((()()(((()()()) (()))()))(()((()))))((()()))((((()()(()((()(()()(((()()()()))())()()))()()()(()()))())))) ()(()()()())((()))))()()()())()(()()((()()()())()))()())((()((((((())((()(())()()()((() ()))(())()(()()()())((()))(((()((())()()()))((()))))()()))(()()))())(((()(()()((((((()() (((((((()((()(()((((())()(()()()((((()))()()(()(()))(()()))()(()(((())()()()())(((()()(())) ((())(((()()((((((())()((((()))((()((((((((())((()))))(((()()(((()()(()((((()()))))(()) ())())(()))()))))(())()(()((()))()()))))())(((((()))))))((()()))()(((()()((((()()(((()()) ()))()(()((())()(()))())))())()(()()()((((((()(())()()(()))()()()())())))()()))())))((()(() ((())()))())))((()()))((((())()))()(((()()()))())(()))((((())((()()(()()))))(()))(()((((()((() ()))()(()(()((())(())(())(((())((()()()()()())()(((()))((((())()()()(()()))))()()())()()(((() ()(()())()(()())()(()()(()()))()()()))((((()(((((()(()))((()()(())))))()))()))(())()() ((((((((()((())())))()()))())))((()))))()()))())))((()(()())()(()())()()))(()()()()()))))() ()()((((((()()()()(()(())()))()()())()))(()(()))))))())()())(())()((((((((()()))))))(()) (())((((((((()())))()()))((((()(())(((()((()(((()()()())(()())))))))()()()((((()(())))) (((()))((((((((((((()()(()))))()()()())))))))())(()))))()(()))(())((((()))(())()))() (((()))((((()))())))(())()()(()(()((((())()())))())(()()()))))))(()(()())(()((())()(((() (()((())(())(())))((((()(())((((((())()(()))())())()()))()(()()())(())((((((())()())) (()))(()))()()))))))()()()()()()(()))()(()(((((()())()(()()(()()))(()()()()()()(()))())()) ())()(()()(()()(()(()()))(()))())(()(())((()))))((()(((()())(()))(()()(()()(()))()() ())()()()))())(()()(())()())()))())()))))())))((((((()((((((((()))()(((((((()()()())))))))) ((((()(((()(((((((((()((()()()())((()()))())()))()()()()))()()(((()((())))))(((()())) ((((()))((()((((())())))()()()))))()()()))))((()()()(())()(((())(()((((((()())))()))()() ()()))))))((()(())()()(())))())()()()))()())()((((()())(())()()))()(((((((()()))()()()() ((())()())()())((((((()())()()(()))()))))()()())()()))(()((()))()()()()(()))()()()())()()() ()))(((((((())))))()()(())(((()))))((())))(()))()()))(()(()))((()))()))(())))()()(()()(() ((((())))))()()(()(()(())()()()))()())())()(()))))(()((((((()()((())()()((((((()()()) ((())))((()()()(()))((()()))()((()((())))))))))()(((((()()()()()())(()(((()((()))((()()))(()) ((((())()()))()()()())(()((()()))))()((((((())()(((((()()(())()()((()((()))((()())))(()) ((()))))()()(()()(()()(()()(()())()(()))))()))))()()(()()(()()()))(())()((()((()))()(() ())((())()((((())()())(((((()()()(()))()()(()(()()))()()))(((((()()()((()()()()(((((())((()(((()()()(()(()))()))()(()()())(((((()))))(((()()(()()()))((()())()()()))(((()() ()((())((()))))(()()()))))))(()))((((()))))))()()(()))))((())(()))()(()()()(()()()))))(()) (((()()())((()((()))(()()())()()())(()()()())(()))))()()()()()()()()((())()()()()()((())) ((((()))(()()()))((()()))()()((())(((()()))))())())))))(((()(((((()((()()()()()()(()))(() (()))))(()(())((((((()()()))()(((((((())()))((((()((((()()()())((((()()()()()()()()(() (((()))()()))))(())()()(()()(()))))))(()()()))()((((()((())))))(()()(()()()((((()()()))))()() (())((()())()()((((())((((()()()()()()()((()))((((()((()()(((((()()()()(()()()()((((((()))) ()(((((((((())()()()()(())()((()()()((((((())))))((()()()()()()()()()()(()))))()()(((()) ()((((((()(())))))))(())(()()()))((()((((((((()(((((...snip... ()))()((()()))(()()((() ()()(()))((((())()()))()))()))((())()))))))))((((((()()(()()()())(()))()()))()())()()((((() ((((()(()))()())()))())()((((())(()()())))(()))((()()()(())()()()()()(()()()())(((()(() ()))(()()(())(()))()())()((()))(((()))(()(()((((()(())(()()))))(((()())(()((((((()()))() ((()))))(())(()))(())()))(()))())()(()()((()))(()))()(()))))(())(((()()()))))(((()(())))) (())((()))((((()()((()(()()))())()(())()()((((()()()))()(())()())())()()(())(((()()(((() ()))())(())((()())(((((((((((()()()(((()()()()((((()()(((()(((((()(()((()(((((()()())) ((()())()))))()()()(())((()(()()))))(((()))))))((((())()()()()()(()))(())(((()(((())))) ()((((((()(((((())()))))()()(())()(()()())()()()()()()(()))((((((((()()()(()((((((()((()))((())) (())(()())()(()()))))))(()()))())(())()(()(())((((()()(()(((((()((((()())))))(()()()()() (()))((((()))))(()))()())()(((((())(()())))))))())((((()((((())()()((()()(()()(())))()() ((()()(((()()()()())()((((((((((((()()(()()((()()()(())()(()(()(((()))())))(()))()()) ()))))((()()((((()(((((()()(()()()((()()()()))()((((()()((()((())((((((((()))))) ()))))()()())(())()()))))))(((((())((((()))))))(())))())(())()()()())))()())))()(())()()) ()))()()()(())((())()))(()()()))(((((((())((()(((()(((((()))()((((()()(())()(()()(()()(()() ((()(((()(((()()(())))(()))()(()((((((((((()((((((()((((((()()()()())(()()()))))()())(() (()()())((()()()(()))((()()(((()((()))()))))()))(()((()))((((()(((((((()()(((((()()()())(()) (((()()))()((()))()()()(()((((()((((()(()()()()))()(()(((((((()))(())()()()))((()()(())((()(()) (())((()))((()()(())(()((())())(()))()(()()))()()()()()))()(()(())))))())(((((((())()))()()) ()(((()()((((()()())))()(()()()()(())((())()()()))()())()))()()))((()())()())((()(()))((()()	82	82	✓

	Test	Expected	Got	
	()(()))))((((()))(()(((()())))((()()()))))))))())(()(((()()))))))))))()()))()())()((()() (()(()((((((()))(()(((((()(()((((((((()()()(()((())((())()((()))(()((())((((()))))(((()) (()))())((((((())))))())((((()((((((()))))))))()((((()))()()((()())))()())))((((()()))()() ((()))((()))()(()))))()))(()()))((()))(((((()((()()())()))))()((((()))()()((((()()))()(()(())))) (((((()(()((())()))()())((()())()((()()))((((()((()()())((()))()((()((((()())))()))))()((()) ())(((((())))()((()))((()())))())))((()((())()))()()()()((((())))()(((((()((((())))) ()(()(((((()()))))(())()((()()))(()(((((()())))(()()()))))))()()((((()()(()()()()(()))))(() (((()()))(()()((())()))))))()(()()((())))()()((((()())))()))())()()(()()))))))()((((())()()() ((((((())))()))))()))((()))))()(()))(())()))((()))()))((()((()(()))()()()))))(()(()))))(())) (((((()(()()))()((()))((())((())(((()(()())()(()(()(((()())()(()()())()()((((((()(()((()))())) (()))(()())((())()))((()()()))))()))((((())))())))((()()((()()(((())((()()()))((()())))) ()))(()(()())(()((())())(()((()))))))(()))))))())((())))()((()(()((((((()())()())(()()) ((())(())()))))())())(()()(())())(()()((()))(()())(())(()))()())((((()())()()()())))) (())))))()(()))((())))((()((())()((((()(()(((((()((())()()()))))()()(()))()()()()()(()())))) (())(()()((())))(()())(()(()((((()))))))((())((((()())))())())())((((((()()())(()))())))(((() (((()())(((((())))))())()(()((())((((()()))((())()()()(()))))()())(((()()(())()()()(()()))))))()) ()()))()))())(()()))))())(((()(())(((()())(()()(()((((()(()(()))))((((((()()(()))()(()()((()((() ()()))()()((()())(()((()()))(((()))()()(())(()))))(())(()(()()((((()((((()(()((((((((((())))))) ()))(()(()(())((()))(()))())(())()()(()()())())()(()))))()()(((()(())(()()()))))))()()((() (()))))()))()())()((((()())(()())(((()()()()))((()))()())())))())))())))(())()))))))((()() ())()))()))()()(()()(()()))(()))))(()())(((()()()()))()()((()))(()(()())()(()()()))()()) ())()))(())))((((((()(())())()))(()()))))(((((()()()())))()()))(())))(())()(((())()()() ((((())()(()((((()())()))(()((()))(()(()()))()(()()))()(((()(()())()))()()))()))(())()(()) ((())(()()()()()()))))()()(())(((((()()())))((((()())))())))))(()(((())(()(()))))))(()())())) ())))))())()(()(())(()))))((()()()))())(((()(()())(())())(((()(()(())))(()(((()()())))) ()))((((())))))(((((())()(())((((())())");			

	Test	Expected	Got	
✓	`cout << mininumBracketAdd("))()((((((())(((()(((((()))((((()(((((()))(()))()(((()))()))))))))(())` 	278	278	✓

	Test	Expected	Got	
	((()(()))()()((()())(())(()()))((()))((((()()(((()))()())))))))))())(()()())(()((()))((()() ()))))()(())()()()()((((()))))(()))))()))((((()))(()()()))))((((())())()())((((((((())))))) (((()))(((((()(()))())()((((())))))()()((((((()()()))(()()((()((()))()()()()()()()))(((((() ()()()))())(())(()((((()(((((()(()))((((()((())()()((((())(()))())())(()((((())()()()))))(()) ())((((())()((()(())()(())()())())((((()))(()))))))((()())(()))(()(())(())()))(()()))))()()) (()))))())())()()()()(((()((((()(()(()))((())(()))())()()((())()))(())((((((())(((((((()) ()))(()()(((((()()((())(()(()((((()((())(()()(()()(())(()()))(()(())()())(()((()))()((((((() (()(((()))((()((()()())((((()(()()))))()()(()))))()())()))((())()(()))))()))))))()()()))(() ((()(((()((((())(((()))()))))))))))(()()())())()(()()(())()((()((())()((()(())(()))())())) ()()()((()(()()(()()(()))(()((((()()))((()((()()()))()()(())(())()()()))))()))()()))))()(()() ((((())(((())(())()()((((()))((()))))((()))(()()())()()(()(())()(())()()))))))(()((())() (((()()(())(()))))()))()()))()(()))))))(()()))(((((((((()))))))))()(())(((()(()(())()))))(())))) (()))))))()))((((())(()()()))())(()))))()((((())()()(()()))))(())()((((())())(()))((((() (()))))))(()))))(()))))()))))))(()))()(((())))))())(())((())()(()(()))()((()))(()((())())) ((()))(((()((()((()()(()))(())))))))))((()(()))()))(()((((()))(((()))()()()((()))))))))(()) (((()())()))()(()))()(((((())))()(()))(((()((()))(()()))()()()))()()((()(()((()(()))))))))) (()()))()(())((((()(()((()()(())()()))))))(((()(()()(()))))(())(()(())(((()))))()()(((((() (())()())(((()((())(()))())))))))(()))))()())()))(())(((((()))(()))()(())(((((())(())()()(()() (()(())()()))))()()()((()))()(()))))()())()))()())((()))()(()((())(()(((())()()((()(() ()))))))))()))((()))(((((())(()))()(()(()()()((()())())(((((())()))(())())(()))(()())))())) (()))(()(((((((()()()))))(())()(()(()(()())()))((()()()()())()())())((((()(()()((((()(())())()) ())()())()()))()()()())(((()()((((())((((())()())(()))((((((()()()())()(())())()))((((()(() (()))()(()((()()(((()()()()(((()()(()((((((())()))())()(())(()))((()))(()))(()))(() ((((()))))()())())()))))()))()()((((())((((())()(()))(())((()()())()()()(((((((()(()))()))()() ())()(()))()()))((()))()))((()()(()(((()()((())()()()(()(())(()(()(((()()))((((()))()(()(()) ()(()(((()))()))))()(()(((()(()()(()(()))))(()()))(())()))())))()");			

	Test	Expected	Got	
✓	cout << mininumBracketAdd(")()())()))()()()(((()((()()()))))())(()(()(())))))()))()) ((())())())()))))((()()((()((((((()()())()(()))))))(()))((())((()((()))())))))())()))() ((()))(()))((()()()))()(())()))()))()(()())((((()())()())((()())()((((((((()() (()(())())()))()(()()))()()))()))()))((()(()))()()()(())()((((())))))))()))) ((((((((()()))))()()))()))))())()))))))))(()))(()(()))())()(()()))()))()))()()())(() ()())(()()(((((()()))())())()()))((()((()))()()))((()))))(((((()(()()())(((()))()()() ((()()()))()))(()))()))()(()()()))))((()))))))))((()()))))))))))()())((())(()))()))((() (())(())()()())(())()))((()())()))()((()))))((())()((()()())()((()())()()(()(((()())(((() ())((())((()))((()()(()(()()))((()())()(()))((()())())))(()(()))(()(()))()()))((((((() ((()))()()()(()()(()()()))((()())()()(()())(()))))))())())(((()(()((()(()))))))()))))()) ()))((((()((())(((((()()()(()(()))(()))()(()))()(()(((()))(()(()))()(()()))(()))()(())()() ())))(()(()))((()))((())(()((((((()()))()))((((()()))))()))))))(()())()()()))((()))()(()() ()))((((())()))))()())((())()()())((()())((())()))(()))))))((((()))(((()()))(())))(((((()()) ()))()()()))()(()((()))(())(())(()()()()))((())))))())()()()()))())()()))))())()())) ((())()((())()()()))))()))))(()))))(()(((((()()()())()()()((((((()(()))((())(((()(())())) (()))))(()()(()()()))()(())((((())((()(((()())((()((((()()))(((((())((()()()())()))()()(()()) ()((()))(()()((((()((()()))(()(())()()))))(()()(())((()(((((()()))(()()()((((()(()))))()() ((((((()((()(())()((()))()(())()))(()((()()))))()))()()(()()))(()((((()((()((()(()(((((() ((()))((())))()())(()(()(()(())))(()))()((())(((()(()(((()()(((()()(()((())((((()()((())(() (()))(())((())(()(()())(())(()()))))((()((((()(())((()))()))((()))()()((()((()(((()()(()) ()((()((())))))))))))()())))(((()(()))()))()((()))))((()()))((())()())())(())(()(()())())))))) ()((()())())())(()(((()((())()())()))((()((())))(()(()))((()()(()()())(((()((())(())(((() ((()))()(()))((((((()))))))))))))()((()))((()))()))((()()))(()(())((()(()))(()()()(()()) ((()))()))((())()()((((((()()()()))))()))))(()(()((()))()(()()(()))((()))((()))()))))(() (((((()(()))((((()(()()((()(()))()()))))()((()()()()((()))()))(())((()()(((())))(((()()() (((())(()(()()()()))()()()()))))())((()))()))))(((()))((()))(()))((()(()()()()()((((((() ((()))(()()()())()()((((()()))()((((()(()(()(()))()))(())(()()()())(()(()((())())))(()(((((()(((() ((()))(()(()))(((()))())(()((()))()((((()(()(()(())())((()))()())))((((((()()()()))()())((() ())()))((()()))((()))((()))()))()()(()))(()))(()((((()()))()))))()())()))))(()(()(((()()(((()))() (((((((()()(()))))))))((()))()()))()(()))()())(()(()))(()(()))(()()()))()()(()()))))(((()() ((())))()))()))(()))(())()))(())()(())(()))(()))(()))()))))(((()()()()))((()(()))(()()())(() ()())((((()(((()))))()))()))()))((()))()))()))((()(()())((((((((()))))(()(())((()(())))))())))) (()))))((()()()()()())((((()()()()))))()))()))()(()))()))()())()))()()((((()()((((()(())))) (((((((())(((()))(()()))()))))()(((((()((((()()()(((((()()()()((((((()())((((()(()))((((()(()()(() ((())()()))))))()())()))))()()(())()(()))))))(()((())()())(()))((())()()))(()))(()(((()))))())))) (()())())(())())((())()))(())()()())()))(()())(()()()))()(()()))()((((())((()()())()))()())(((() (()))))()))()((((()))(()()())()())((()))))((((((()))(()()()(())()(()))()())((((()))))(()()) ((())()((()((()()(()(())((((()((()())((((()(()(()()((((()()(((())()(())))((((())))))(())()) ((((((()()(((()()(((())()((((()()()))((()((()(((()))())((()(((((()()()(())()(()())((((()))(())) (()()(()()))())((()((((())((()))()())(()))()()(((((()())))))))((((()))(()()())()()(()()))))())) (()))((((()(()(())(()))((()(())))((()))()(()()))(()))))))(()(()))((()(()((()))()()((())) ())(()))()(()()(()))()((((()()))()((()()()))()()(()(())((()()()))(()))()((((()))()))(()() (()((()))(((((((()()()))((((()((((()))(()))(((((((()))()()()((()((()))(()))(()))()))))(() ()))))()(()())(())(((()())((((()((((()(())()))))())((()))(()))(((()((((()(((((()(())(()((((()) ((((((((()()((()))))(()()((())())((()))))((((()))())())((((()()()))((((())())()) ...snip...) ())((((()(()()()))(()(()))())))((())(()()())()((()()))((()))((((())))))())(()(()()())(((()(()()) ()()((()()()()))(()(()()(()()()))))())()(())())((((()()(()))))(()(()))())))()))))())((()() (()))))((()())((((())((((()()))(()((())))))()())))()(((()))))()(())())()())(()))())(())(((() ())((((()())()(()()))()()(())())(()()()((())(()(()())()()()))()()()((())(()))()())((()) ((((((()(((()))(()((()((((((()()))))((()((())))()))()((()))(()()))())()(((((()()))))(() (()()))))()()))()()))()))((()))((((()))(()())((()(())()))()(()))))()((()((())(())(())()() ((((((()))())(((((((()(()))(()(()()))()(()(()((((()((()()()((((((()))))))))))()()(()))())))(() (()()(()(()))(((((()))))(()()((()))())((()()(()()(()))()(()))))))((())(()))(()((((() ()))))()(((((()(())()()((((()(()((()(((()(((()))(((((((()((()()((()()))()()((()())()())() (())((()(()()(((()(()((((()(()))()(())()())()()))))((()((((()()))))()()))()))())))) ((())())(())())(())()()()()(())()))()(()()((()()(()((()())))()()()))((())))())))) ((()))))))(()(()(())((()()))())))((()((()(()(()()()))((()))))(()(()((())((()()())(()) (())()()))((()(((()())))()((((()(())(((()((((((()()(()((((()))(()()((()()))())(()())((() ()))())))(()((()()))()())(()()()(((((()()()((((((()((((()()()()(()((()))(()(()))((((((()(((() ())(((((((()(())((()()()())())(((()((()((())(((((((()()()))(()))((()))((((()()(())))))() ()()()())(((((()))))())())(()))()()(()((())()(((()())))())))()(())()()())(()((()))())))) ())())(()()(((()(()(()((()))))(()))()((()))(())(()(()))))))))))(()))((()))()())))))) (((((())((()))()())()))))()()()(((()((())())))))))()(()))))())(()))(()))(((((()()(()() ((())(()((((((())))(((()((((()))()()))(()))))))()))()()()()))((())()((((())()(((())(()((()	202	202	✓

	Test	Expected	Got	
	()))()))))(((()))(()()())(((((()))))(((()))))(()))))))))((((()()(((((((((((()))))()))))))(()()(())			
	()()()(()))()()()))(()))((((((((((()))))(((()()(((((((()()()()()()()()()()))))))))))((((((((((			
	(((()))()))))(()))))))))))(((((()))))(()()(((()))(((((()))))(((((()))))(((((()))))))))))))))))(((()()			
	(()(()))(((((()()(((((()))))))))(((((()))))))))(()()))))))(((((((((((()))))))))))))))))))))))))(()			
	((((()(((((()))()			
	()()(()))))))))))(((((((((((((((()))()			
	(()(((()()(((()))))))))))(((((((((()))()			
	()()(((()))()			
	((((((((()))()))			
	()))))(((()))()			
	(((()))(((((())			
	()(((()))()			
	()(((()))()			
	(((((())			
	()))))(((()))()			
	(()))))(((()))()			
	()(((()))()			
	(((((())			
	()(((((((((()))()			
	(((()))(()			
	((((()))(((((()))");			

	Test	Expected	Got	
✓	cout << mininumBracketAdd("(()))((((())()((()()())()))()((((())(()))(((((()(())())()((((((() (()))(()()(((((()()())(()())()((((()(()(())(())(()()()((()(())()())()((((((((()()))(())))) ((())()()(()()))()(()(())()))(((((()()()))()))(()())()())))()())()(()())()()))((((()))))) ())((()(((()((((((()))()()))))()()))))()())()((()()))))()))((()(()))))()))))((()()()((()) ()))))()((()()(((()(()()((((()(()()(((()(())))()))((())()()()()))()))(()))))()))))())((((() ()((((())())(((()(()))))))))((())()))(()))))))(()))))())))(())()()()(()))))()()))()()((() ())((()()(())))((()))))(()))))()((()(())))()(())()()))((((())(()))()))(()))))()))(()))))(() ()()))()()))(())(()()((())(()(())()()()))()(())(()()()(()((()()))))(())(((())))()))(((() (()((()((())(()()))))(()()(()()))))(((()()(()())()()(()))()))(()))()()()((())(()())())() ())(())((()()))(()()())((())()(()))()(()))))((((()()(()()(()))((()()()((((()()))))(((()) ()))((()()))(())((()))(()))())())()((()(()((()()))))(()))()()()(())()(())((()()))((()((() (((()((()()(((())((()()(((()()(((((()()()())))(())(((()())()((()(((((()()())(((()()))) (())))))))(())()(()(()))((((()))))(((((()))(())()))(()(((()))))()))(()(()))(())))((((()() ((())(((())((((((()(((())(())))))()())())()))(())))))(((()()))))(())(((((((()(((((()()())))))) (()))))()()))))((((()()(())()))))())()))))))))()()()())()()()(()()(())(()()()()(()))())) ()))())((())(())(()))(())))))()))))(((((((((((()(())(())())))))()))(())(()()()(((() ((((((((((()(()))))())()()))(((())(())((())))()))))((((()()))))()((((())((()()(()()((((((()(((() (()))))))())())())()((((()(()))))())))))))((((()()()))())((()()((((()()))())()()(((())(()) ()((((()(()(())(())((()()()))(()))))(()(()())))))((((()()))((()(((((())()(())((((()))()() ()))()(())())()()())()()((((())())()))()())()((((()())(()(()))()())()())()))))()((((((()((((() ((((()()())()()))()(((()())))()))())(()))(()())(()(()()(((()((((()())())(()))())(())()())))() ((())()(())((()((())(())()()()(((((()))))()()(()())((((()))((((((((()(((((())()))()((((((() ((((()(())(())()))()()()))(((((()((((()()(())(()()))()(())()()()(()()()((()(()))((())) ()((()())())())()))))()(())()())((()))))()))))((((((()(()((()))))((()(()(((((())(())(())())) ()()()((()())())(((((()((((()))(()))(()()()))))(())()((()((()))())()())()()(()(()))(()()(() (()((((((()((((((()(((((()()))))(()()(()))()())()(((()(((((()(()()(()))(()(())()()((((((()()) ()))())))(())()(())))()()((((())()()())()(())()))(()((((()(((())()(()))))()((())()())(()) ()))))))))())(((()))(())(()(()(())(()))())(()()()(((()(()()))((((()(((()))()()()()(())(((()))(() (()()))()()()))(()))(())))()((()))(())(()((((()))(()()))()((()()))))()()((((()(())) ((((((((()())(())(((()()))())))()))(((((()))))((((()))))())))))((()()(())())(())()()(((() (())))))))(()((((()()())()))(()((()()()()()))())()()(())(())()((((()()())()()(()))((() ()()((()))((())((((()))()()))((((()(()((()(())()()(((()(()((()()()(()))()))()))))(()))()))() ()()()))()(()))(((()(()(()(())(()))(()(()))))((((()(())))()()()(((()(()()()()()(()(()() ()((()()())((((()))())))()())(((()((()()()))))))()()))(((()))(()(()(()))()))()(()((((()) ((()))()(()(()))()()))(()()))((((()))()()((()()(((())))()())()()()))((()))))()((()(()))())) ()()())(()((((()))))(()))()((()()))()())()())())(((()((()))((((()(()))((((((()(())(()))))() (()))((()))))))())()(()((((((()(()()()()))()())()()()((((()))()()))))(())(()((((()))()))()())() ((((()((()()))()))))((()))()))())(()((((()()()()()())()())(()((((((()(()))()))(()()(()())) ()))))))())))()())()())(()((((()(()))(((())))))()))())()()))()(()(((()))(()))(((())(())) (())()())(()()))((()()))((())(()()())(((())(((()()))()()()())()))()))(()((()))()())((((() ())()))))(()()))))())(())()))((((()((((((()))))))()))(())((()))))(()))()))(())))()()(()()((() ((((()(())(())()(())(()()))))(()(()()((())(())((((((()))(()()()(())(()))())(()())(()()))(()) ((((((()()))()((()(()(()(()()())()(())(())()))()(()()(())))()(()(()))())))()(((()((()))() ()))((()))()(()(()(()(()()))())))))()(()(()(()))()(((()))))()))()()((()((((((()))()()) (()))())(()()))((()(()()))(()()))))((()(()())))())((((()(()()(()))()())())))(()(((((()((...snip... ()()()((()(())())())()))((())((((()()())()((())()))(())(()()))()))()))))())))))) (()()()))((()))()(())(((((()()(()()()))((()))()())(((()(())((()()()())())((())()()))))))(()) ((())()((((()(((()((()))((((()(())()))()((((()(()((()(((()(())))(()(())()()(()))(()))(())) ())((()))()((((()()())()))))()(())())(()(()()))()((((()(()()())()))(()()))()()(((()(((())) (((()))))))())())()))))()))())(()))()((((()((()))(()())()))()()(()))))(()()))()(()())))) ((())))))()(()((()(((()))(((()(())()(()(())()(((((((()()()(()()(())((()()))()()()))(((()) ((()()(()()((()((((())())))(()()(())()(()()(()()))))((((()(((()())))((((()()(())())))()()(((((() ((((((())(((()((()))()))()))))))())))((((((()()())()((((((())())()(()(()))(()()()()()))(())) ()))()(()()((((()())))))())(((((()()))))((((((())(()(()(()))()(((())))()))(()()()()())(((()) ()()((()()))))()()(()((())((()())()((()())()(((())(((((()(()))))()(()()))()())(((()(((() ((()((((()))()()())()))))()()()(())(()()((((())())(()())()()()((((((())()))()()(()())))))) ((())(()())()(())(((()))())((((()(()()(()))((((()))())((((()(((((()((()))))()(())()())(()())(()) ()))((()))()())(((((((()(()()(())()()))))))(()(()(((()(()(((()((((((())))(((())))()() ())))))))(((()()()())()((((()(((((())()(())()(()(((((())()()()())()(()((((()(()((((((()) (())(()()(()()())()(())))((()))()(()))))()((((()((((((())))))((((()((()()(((()))(()))()) ()))(((()()(()(((()(()(())(()))()(())))((((()(((()(((()((())))(()())()(()(((()(((((()) ())((()())((()))()((()))()(()))(()(()()))((()()((()())()))()))))((()))()((()())()()) ((((((((((((()()())(())())())(())()))()(()())))))())(()))()())()()))))(())))()()((()())()((()) ()()))))((((())(()()(()))((()(()(()())(((((()))((((((()(()()(()))))())))(())(((()()))(()))	426	426	✓

	Test	Expected	Got	
	())()(()))()))()))())(())())()))))())())(()()())()())((()())())((((((((((())((()())())))) ()(()))((()()()(()(((((()((()())()())(()(()()(()())))))()()(())((()()())()))((()(() (()(()))()()(()))(()(()())(()))))()()())(()))(())()()((((()(()())()())())())()(((() ())()())(()))(())))))()))))()(()))())()))(()()())(((((()(()))(()(()((((()))) ())))))))))(()))(()))(())(()(()(()()()(((()(()(()))(()()))((((()()))())()(()())))(()) ()(()))((()()))(()()))((((()(()(()()))(()))(())(())(()((((()()((()())))(((()))))() (())()()(()))(()())))))()()(()))))(()))(()()(())))()))))()()))))()))(()(()))() ()))(())))()())((())(()))()(()()))()(())()))(())()(()())))()()))))()()())))()((())) ())((((((()))(((()()()))())(()))()()()())))()))(()()()(()((())()()()))(()()()()))))) ()))()())((()((((()(((()))((((()((((()((())()))))())(()))()(()(()((((())))(()))(()))((())()) (()))()()(()((((()()(()))())(()()()()(()(((()(()(((()))()(()))())()((((((()()(()))(())) (((((())(()))(()(()((()(((()(((((())))((())(())((()))(()))()()(()))(()(())(()))))()())(() ()))(()))()()()(()())((((()))((()))(())(()))((()())()()((()(()))()))))(())((((()))())))) ()))))))))()(()())(((()(((()((()(()()))()()(()(())(()()(()()(()((((()()(()()(()()())) (((((())))()()(()()()(()))(((((((()))())(()((()(())))())))()(()()((()())(()((((()((()((() ((())()()))))()(()))()(())()(()((((())()))))(()())((()((())(())((()(()(()((()))())(()()(() (())())(()((((()(()(()))(()))(()(()()())((()()(()))(()(()())()))))()((()(((((())()() (())()(()((()))()(()))((((((()))())(())((()(((((()((((()))()))())))()()((()((()((((()()())))) ())((()()(())(()))()(()))()(()(()())()()))(()()()()()())(()))))(()()(()))(()((()))()) ((((()))())(()()(()()()))((((((())(()))))((()(()))(()(()))))(()))))()(()))))())(()))()))()) ()((())))))()))(()))()())()((())(((())((()))))))))()(((()))((()(()()()(()))())()()(())) (((())()))((())((()((()(()())(()()()((()(())()(())((((((()()()(((()()()()()())(((((((() ((((()(()))((())(()(())((())((()())((()))((())(())((())(((()((()))(()))))(())(())())(())) (())((((((()()(()))())(()))((()((()((()(())((()((()()))()()())()(()())((()(((((()()(()())) ())))())(()())((()))))))))))(()()(()((())(()((()((()((()))))()(()))()))((()(()((()() ())())((((()))(()()((()())(())))(()()()(()))()((((()()((()))()))))))))))(((())()))(()(()())) (");			

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **11**

Đúng

Đạt điểm 1,00 trên 1,00

Given a string `s` representing a sentence consisting only of `a-z` and `A-Z` and space character.
Your task is to implement a function with following prototype:

```
string reverseSentence(string s);
```

The function returns the reverse sentence of sentence `s`.
The testcases ensure that there is only one space character between two adjacent words, and the sentences do not begin or end with any space characters.

Note:
- The `iostream` library has been used and `namespace std` is being used. No other libraries are allowed.
- Using loop keywords (`for`, `while`, `do`) are not allowed, even in comments and variable names.
- You can write helper functions.

For example:

Test	Result
`cout << reverseSentence("data structure and algorithm is scary");`	`scary is algorithm and structure data`

Answer: (penalty regime: 0, 0, 0, 5, 10, 15, ... %)

Reset answer

```
// Helper function to find the first word and the rest of the sentence
pair<string, string> splitFirstWord(const string &s) {
    size_t pos = s.find(' ');
    if (pos == string::npos) {
        return {s, ""};
    } else {
        return {s.substr(0, pos), s.substr(pos + 1)};
    }
}

// Helper function to recursively reverse the sentence
string reverseHelper(const string &s) {
    // Base case: if the string is empty, return an empty string
    if (s.empty()) {
        return "";
    }

    // Split the first word and the rest of the sentence
    pair<string, string> parts = splitFirstWord(s);
    string firstWord = parts.first;
    string rest = parts.second;

    // Recursive case: reverse the rest of the sentence and append the first word
    string reversedRest = reverseHelper(rest);
    if (reversedRest.empty()) {
        return firstWord;
    } else {
        return reversedRest + " " + firstWord;
    }
}

// Function to reverse the sentence
string reverseSentence(string s) {
    return reverseHelper(s);
}
```

	Test	Expected	Got	
✓	`cout << reverseSentence("data structure and algorithm is scary");`	`scary is algorithm and structure data`	`scary is algorithm and structure data`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.

Câu hỏi **12**

Đúng

Đạt điểm 1,00 trên 1,00

Given a string, implement function

```
int strLen(char* str){}
```

to calculate length of the string using recursion.

Please note that you can't using key work for, while, goto (even in variable names, comment).

For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`char str[] = "Truong DH Bach Khoa";` `cout << strLen(str);`	19

Answer: (penalty regime: 0 %)

Reset answer

```
1
2
3  // Function to calculate the length of the string using recursion
4  int strLen(char* str) {
5      // Base case: if the current character is the null terminator
6      if (*str == '\0') {
7          return 0;
8      }
9      // Recursive case: move to the next character and add 1
10     return 1 + strLen(str + 1);
11 }
12
13
```

	Test	Expected	Got	
✓	`char str[] = "Truong DH Bach Khoa";` `cout << strLen(str);`	19	19	✓

Passed all tests! ✓

Đúng

Marks for this submission: 1,00/1,00.