Instructions

This assignment is designed to help you practice basic C++ concepts. There are 12 questions in total, including multiple choice questions and coding exercises. The questions aim to improve your understanding of C++ fundamentals like input/output, variables, data types, and selection structures (if, ternary operator, and switch case). Therefore, cheating is forbidden.

I. Multiple choice questions:

1) What will be the output of the following code?

```
int arr[] = {1, 2, 3, 4, 5};
int sum = 0;
for (int i = 0; i < 5; i++) {
    sum += arr[i];
}
cout << sum;
```

a) 10
b) 12
c) 14
d) 15

2) What will be the output of the following code?

```
string str = "Hello";
for (int i = 0; i < str.length(); i++) {
    if (str[i] == 'e') {
        str[i] = 'a';
    }
}
cout << str;
```

a) Hallo
b) Hella
c) Hille
d) Hullo

3) Which of the following statements about recursion is true?

a) Every recursive function must have a loop inside it.
b) Recursive functions can be used to replace any loop.
c) A recursive function must have a base case to prevent infinite recursion.

d) Recursion is generally more efficient than loops for all problems.

4) What does the following code output?

```
int arr[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}};
cout << arr[1][2];
```

a) 2
b) 4
c) 5
d) 6

II. Coding problems:

1) Write a C++ program to implement a recursive **digitsSum** function to calculate the sum of digits of a given number.
2) Write a function **countVowels** that takes a string as input and returns the number of vowels (a, e, i, o, u) in the string.
3) Write a function **transposeMatrix** that takes a 3x3 matrix as input and prints its transpose. The transpose of a matrix is obtained by swapping rows with columns.

Example:
For matrix:

[1 2 3]		[1 4 7]
[4 5 6]	Transpose is -->	[2 5 8]
[7 8 9]		[3 6 9]

4) Write a function **findMax** that takes an integer array and its size as parameters and returns the maximum element in the array. Implement a simple main function to test it.
5) Write a function sumDiagonal that takes a 3x3 2D array of integers and returns the sum of its diagonal elements. Test it with a sample array in the main function.
6) Given the array of integers nums, you will choose two different indices i and j of that array. Write a **maxProduct** function that returns the maximum value of (nums[i]-1)*(nums[j]-1).

Source: LeetCode

Example 1:
Input: nums = [3,4,5,2]
Output: 12
Explanation: If you choose the indices i=1 and j=2 (indexed from 0), you will get the maximum value, that is, (nums[1]-1)*(nums[2]-1) = (4-1)*(5-1) = 3*4 = 12.

Example 2:

Input: nums = [1,5,4,5]
Output: 16
Explanation: Choosing the indices i=1 and j=3 (indexed from 0), you will get the maximum value of (5-1)*(5-1) = 16.

Example 3:
Input: nums = [3,7]
Output: 12

7) Bonus

- Write a function generateArray that takes an integer size and returns a dynamically allocated array of integers. Each element in the array should be set to its index multiplied by 3. Include a main function to demonstrate usage.
- Write a function applyFunctionToArray that takes an integer array, its size, and a function as parameters. The function should apply the passed function to each element in the array. Write a sample function that doubles each element and passes it to applyFunctionToArray.
- Write a function to slice an array from a given start index to end index.
    - Slicing refers to the process of extracting a subset of an array from one index (start) to another index (end).
    - Given an array and the indices to slice at, return the sliced array.
    - For example, if arr[] = {15, 7, 13, 20, 9}, start = 1 and end = 3, the expected output is {7, 13, 20}.

  The function definition:

```
vector<int> sliceArray (vector<int>& arr, int start, int end) {   }
```

- Write a function to add an element to a nested array.
    - Add an integer to the end of each sub-array.
    - Return the updated nested array.
    - For example, if arr[] = {{1, 2}, {3, 4}, {5, g6}} and num = 7, the expected output is {{1, 2, 7}, {3, 4, 7}, {5, 6, 7}}

  The function definition:

```
vector<vector<int>> addToNestedArray(vector<vector<int>> arr, int num) { }
```