

# FAST NUCES Programming Olympics

## PROGRAMMING FUNDAMENTAL CS1002 - Fall 2024

Release Date: 11th October 2024

Due Date: 29th October 2024

Total Marks 90

Congratulations, athletes! You've advanced to the next stage of the Programming Olympics for Fall 2024 by excelling in Assignment 1. Your journey in CS1002 continues with the second round of challenges.

- The rules remain simple: Your mission is to cross the finish line, but not just with speed. This time, we focus even more on elegance, creativity, and the dedication you put into your problem-solving journey.
- True champions face obstacles with integrity. Remember, the biggest challenge isn't just solving the problems—it's overcoming the temptations of using shortcuts like ChatGPT, Gemini, or relying on others' solutions. Such actions will lead to immediate disqualification.

Now, show us your best work, stay honest, and keep your eyes on the prize. Let the challenge begin!

## How to submit

---

There are a total of 9 questions in the assignment and each question carries equal marks i.e 10. You must submit the .c files compressed in a zip folder with your student id on google classroom within the due date.

## Question : 1

---

Write a c program to find the second smallest element in an array.

Input 5 elements in the array (value must be <9999) :

element - 0 : 0

element - 1 : 9

element - 2 : 4

element - 3 : 6

element - 4 : 5

*Expected Output :*

The Second smallest element in the array is : 4

## Question : 2

---

A popular beverage company is looking to optimize their marketing campaigns by understanding the frequency of characters used in their promotional slogans. They believe that analyzing these patterns can help them tailor their messaging to specific demographics and improve brand recall. You've been tasked with creating a C code function to analyze the character frequency in a list of slogans provided by the marketing team. This analysis will help identify the most common letters used in their slogans, which can inform future marketing strategies.

### **Input:**

slogans = ["buy now", "save big", "limited offer"]

### **Expected Output:**

- For "buy now": {'b': 1, 'u': 1, 'y': 1, ' ': 1, 'n': 1, 'o': 1, 'w': 1}
- For "save big": {'s': 1, 'a': 1, 'v': 1, 'e': 1, ' ': 1, 'b': 1, 'i': 1, 'g': 1}
- For "limited offer": {'l': 1, 'i': 1, 'm': 1, 't': 1, 'e': 2, 'd': 1, ' ': 1, 'o': 1, 'f': 1, 'r': 1}

## Question : 3

---

You are working for a text-processing company that optimizes storage for large amounts of written data. The company has identified a recurring issue: words in documents are often unnecessarily lengthy due to repeated consecutive characters. Your task is to minimize these words by reducing consecutive duplicate characters to a single instance of the character.

### **Operations:**

**Compress a Word:** Write a function that takes a string (a word) as input and returns a minimized version of the word, where all consecutive duplicate letters are reduced to one.

**Multiple Word Compression:** You will then apply this function to a list of words and return the minimized version for each word.

Add a feature to calculate how many characters were removed during the compression process.

Input: words = ["boooooook", "coooooool", "heeeey"]

Output: ["bok", "col", "hey"]

## Question : 4

---

You are working as a software engineer at a company that manages a large database of customer transactions. Each transaction is stored as a string that contains encoded information about the customer. However, some transactions may have been duplicated due to system errors, and these duplicates appear in scrambled order. Your task is to identify and group these "scrambled" transactions together, as they represent the same customer activity. Two transactions are considered scrambled if they contain the same letters but in a different order (anagrams). You are provided with a list of transaction strings. You need to write a program that groups these scrambled transactions into separate categories, returning each group of related transactions together.

Example

Input transactions = ["eat", "tea", "tan", "ate", "nat", "bat"]

Expected Output: [['bat'], ['nat', 'tan'], ['ate', 'eat', 'tea']]

## Question : 5

---

You are tasked with creating a program that generates a histogram based on user input. The user will provide a set of values, and your program will use loops to create a visual representation of these values.

1. **Input Handling:** Write a function that takes an array of integers (the values) and the count of those integers as input.
2. **Horizontal Histogram Function:** Use loops to generate and print a histogram, where each value is represented by asterisks (\*).
3. **Vertical Histogram Function:** Use loops to generate and print a vertical histogram, where each value is represented by asterisks (\*).

**Expected Values:**

```
int values[] = {3, 5, 1, 4};
```

```
int count = 4;
```

**Horizontal Histogram**

```
Value 1: ***
```

```
Value 2: *****
```

```
Value 3: *
```

```
Value 4: ****
```

**Vertical Histogram**

```
  *
```

```
 *  *
```

```
* * *
```

\* \* \*  
\* \* \* \*  
3 5 1 4

## Question : 6

---

Two friends, A and B, are playing the game of matchsticks. In this game, a group of N matchsticks is placed on the table. The players can pick any number of matchsticks from 1 to 4 (both inclusive) during their chance. The player who takes the last match stick wins the game. If A starts first, how many matchsticks should he pick on his 1st turn such that he is guaranteed to win the game or determine if it's impossible for him to win? Write a function which returns -1 if it's impossible for A to win the game, else return the number of matchsticks he should pick on his 1st turn such that he is guaranteed to win.

## Question : 7

---

One of the master coders designed a subscript block that stores random characters in a multidimensional array. You are asked to design a program that will find a given string in a multidimensional array of characters. The search for characters can be present and operational from left to right and top to down only. The program should create a 6 x 5 2D array and populate it with random alphabet characters. After that, the program should print it in a tabular form as shown below. Search the user-entered string in the 2D array, if it is present then add a point to the score, if it is not available then subtract one. Print the score at every input. The program stops asking and re-populates the 2D array with new random characters when the user enters "END" as the string.

[NOTE: After generating random numbers, the last row should have your student ID's last four digits]

E	D	D	F	R
A	F	V	A	Q
T	E	B	S	T
L	J	G	T	T
1	2	3	4	Q

Search Str= "FAST"

Output:

FAST is present Score 1

Search Str= "EAT"

EAT is present Score 2

Search Str= "GREAT"

GREAT is not present Score 1

## Question : 8

---

Write a C program that contains the following functions for different numerical conversions:

1. `int BinaryToDecimal(int number);` Converts a binary number to its decimal equivalent.
2. `int DecimalToBinary(int number);` Converts a decimal number to its binary equivalent.
3. `void DecimalToHexadecimal(int number);` Converts a decimal number to its hexadecimal equivalent and prints it.
4. `void HexadecimalToDecimal(string hexNumber);` Converts a hexadecimal number to its decimal equivalent and prints it.
5. `void BinaryToHexadecimal(int number);` Converts a binary number to its hexadecimal equivalent and prints it.
6. `void HexadecimalToBinary(string hexNumber);` Converts a hexadecimal number to its binary equivalent and prints it.

Each function should take an appropriate input and return or display the converted value. Ensure that the program handles invalid inputs gracefully. Your program must display the menu which function the user wants to call.

## Question : 9

---

Write a C program to simulate a simple grid-based adventure game using a 2D array.

### Game Description:

- The game consists of a 5x5 grid.
- Each cell can contain:
  - An empty space (' ').
  - An item ('I') that the player can collect.
  - An obstacle ('X') that the player cannot pass through.
  - The player's position, represented by 'P'.

### Example Input:

```
char grid[5][5] = {  
  
    {' ', ' ', 'I', 'X', ' '},  
  
    {' ', 'X', ' ', ' ', ' '},  
  
    {'I', ' ', 'X', 'X', ' '},  
  
    {' ', ' ', ' ', ' ', ' '},  
  
    {' ', ' ', ' ', ' ', ' '}
```

```
{',', ' ', ' ', ' ', 'I', 'X'},
```

```
{',', 'X', ' ', ' ', ' ', 'P'}
```

```
};
```

**Requirements:**

1. Display the grid and prompt the player for a move (W: up, S: down, A: left, D: right, Q: quit).
2. Update the player's position based on valid moves.
3. Collect items when the player moves to a cell containing an item (remove item from grid).
4. Prevent movement into obstacles ('X').
5. Continue until the player chooses to quit.
6. The grid is fixed at 5x5.
7. Only valid moves are allowed.
8. Handle invalid input gracefully.