



COMSATS University Islamabad
Department of Computer Science
Terminal Examination - Fall 2024

Class: BSSE 2A

Subject: CSC103-Programming Fundamentals

Instructor: Hafiza Sundus Farooq

Marks: 50

Time allowed: 3 Hours

Date: 07-Jan-2025

Instructions:

- All questions are self-explanatory and require no further explanation during exam time.
- Return the question paper along with the answer sheet.

[CLO-1] Demonstrate the fundamental concepts of programming.

Question# 1:

(6 marks)

Show the output of the following code snippets.

i.	<pre>public class A { public static void main(String[] args) { int base = 3, exp = 10, mod = 5; int result = (int) Math.pow(base, exp) % mod; System.out.println("Result: " + result); } }</pre>	ii.	<pre>public class A { public static void main(String[] args) { int x = 100, y = 30, z = 7; int result = x / y * z % y + z; System.out.println("Result: " + result); } }</pre>
iii.	<pre>int id = 101; String name = "John"; double score = 89.657; System.out.printf("ID: %3d Name: %10s Score: %.1f\n", id, name, score);</pre>	iv.	<pre>public static void main(String[] args) { short value = 32767; value++; System.out.println("Short value : " + value); }</pre>
v.	<pre>int u = 3; int v = 7; int w = 18; int result = ++u * v-- - (w++ + --v) + (u++ - w--) * ++v; System.out.println("Result: " + result); System.out.println("Values after expression: u = " + u + ", v = " + v + ", w = " + w);</pre>	vi.	<pre>public static void main(String[] args) { String str = " 456 "; int num = Integer.parseInt(str.trim()); System.out.println("String converted to int: " + num); }</pre>

[CLO-2] Employ programming constructs using a programming language.

(8 marks)

Question# 2:

Write a method that returns a new array by eliminating the duplicate values in the array using the following method header:

```
public static int[] eliminateDuplicates(int[] list)
```

Write a test program that reads ten integers, invokes the method, and displays the result. Here is the sample run of the program:

Enter ten numbers: 1 2 3 2 1 6 3 4 5 2
The distinct numbers are: 1 2 3 6 4 5

(9 marks)

Question# 3:

Suppose you are working for a small logistics company that manages its inventory manually. Your task is to help the company automate a part of its inventory management process by writing a Java program. The inventory consists of multiple items, each represented by a unique item code and a quantity. You are tasked to implement a program that can: 1. Search for an item in the inventory using its unique code. 2. Update the quantity of an item based on its code. 3. List all the items in the inventory recursively. Here's the sample run of the program:

```
Please choose an option (1-Search, 2-Update, 3-Print, 4-Exit): 1
Enter an Item: A101
Item found: A101, Quantity: 50
Please choose an option (1-Search, 2-Update, 3-Print, 4-Exit): 1
Enter an Item: A110
Item not found
Please choose an option (1-Search, 2-Update, 3-Print, 4-Exit): 2
Enter an Item: B202
Enter new quantity: 40
Quantity of item B202 updated to 40
Please choose an option (1-Search, 2-Update, 3-Print, 4-Exit): 3
Inventory List:
Item: A101, Quantity: 50
Item: B202, Quantity: 40
Item: C303, Quantity: 10
Please choose an option (1-Search, 2-Update, 3-Print, 4-Exit): 4
```

Question# 4:**(8 marks)**

Given the following *AddressBook* class, complete the *insertContact* function, which will insert a contact into the address book in sorted order. The sorting should be maintained based on the *name* attribute. Insertion at the start or in the middle must shift the array down to make space for the new contact. Finally, write code in main calling *insertContact* to insert a new contact in the address book.

```
public class AddressBook {
    static final int ADDRESSBOOKSIZE = 1000; // Address book size
    static String[][] contacts = new String[ADDRESSBOOKSIZE][4]; // Stores name, phone, email, notes associated with the contacts
    static int contactsCount = 0; // To keep the current size of the Address Book
    public static boolean insertContact(String[] contact) { // Inserts a new contact in the Address Book in a sorted way
        return true; // Returns true if the contact inserted successfully and false if is no space left in the Address Book
    }
    public static void main(String[] args) {}
}
```

Question# 5:**(9 Marks)**

A social network can be represented in the form of a binary matrix, where the element in row *i*, column *j* indicates whether user *i* is friends with user *j* (0 -> not a friend, 1 -> friend). For example, the following network of five users tells us that:

0	1	1	0	0	1
1	1	1	0	0	0
0	1	1	1	1	1
0	0	1	1	1	1
1	0	1	1	1	1

- User 0 is friends with Users 1, 4
- User 1 is friends with Users 0, 2
- User 2 is friends with Users 1, 3, 4
- User 3 is friends with Users 2, 4
- User 4 is friends with Users 0, 2, 3
- User 0 and User 2 have two mutual friends
- User 2 and User 4 are social butterflies

Given the following *SocialNetwork* class, complete the following functions:

- *addUser* to add a new user to the network
- *findMutualFriends* to find IDs of the mutual friends between two users
- *findSocialButterfly* to find ID(s) of the user(s) with the most friends

```
public class SocialNetwork {
    static int noOfUsers = 1;
    int[][] network = {{1}};
    public static void addUser() {} // Adds a new user to the network by increasing its size to accommodate the new user
    public static int[] findMutualFriends(int user1, int user2) { // Finds all the mutual friends between two users
        return null;
    }
    public static int[] findSocialButterfly() { // Finds the user(s) with the most connections
        return null;
    }
}
```

[CLO-3] Handle programs exceptions, and file I/O.**(10 marks)****Question# 6:**

Suppose you are working on a blog editing platform that allows writers to upload their drafts in text files. To enhance the editing tools, you need to provide writers with a quick summary of their content. Specifically, you want to count the total number of characters in the draft, the total number of words, and the total number of lines.

The draft files will be uploaded in plain text format (.txt). Words are separated by whitespace characters, and each new line represents a separate line in the draft. Write a program that reads a file named *BlogDraft.txt* and generates a report summarizing the number of characters, words, and lines. Handle any potential issues, such as the file being missing, and write the results to a file named *DraftSummary.txt*.

BlogDraft.txt

```
This is a blog draft.
It has multiple lines and words.

Let's count the characters, words, and lines!
```

DraftSummary.txt

```
Content Analysis Summary:
Total Characters: 101
Total Words: 18
Total Lines: 4
```