ourdiu.com | Shared by Sumit Saha

Daffodil International University

Faculty of Science & Information Technology
Department of Computer Science & Engineering
Final Examination, Fall 2025
Course Code: CSE113, Course Title: Programming and Problem Solving
Level: 1 Term: 2 Batch: 69

Time: 02:00 Hrs **Marks: 40**

Answer ALL Questions

[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1. **Identify** the errors in the following code? **Explain** the errors and reasons why you think they are errors. — CO2

a) — 03

```
1. #include <stdio.h>
2. void increaseScores(int *arr, int n*) {
3.     for(int i = 0; i <= n; i++) {
4.         *arr[i] = arr[i] + 5;
5.     }
6. }
7. int findMin(int *p, int *q) {
8.     if(p > *q)
9.         return q;
10.    return *p;
11. }
12. int main() {
13.     int scores[3] = {10, 20, 30};
14.     increaseScores(scores, 3);
15.     int result = findMin(&scores[0] &scores[2]);
16.     printf("Minimum score is: %d\n", result);
17.     return 0;
18. }
```

b) **Rewrite** the code without any errors. — 03

2. **Generate** the output of given codes below (write only the output segment in a box): — CO3

a) — 03 + 03

```
#include <stdio.h>
int main() {
    int numbers[3] = {4, 7, 2};
    for(int i = 0; i < 3; i++) {
        if(numbers[i] % 2 == 0)
            printf("%d is even\n", numbers[i]);
        else
            printf("%d is odd\n", numbers[i]);
        if(numbers[i] > 5)
            printf("%d is greater than 5\n", numbers[i]);
        else
            printf("%d is not greater than 5\n", numbers[i]);
    }
    return 0;
}
```

b)

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
void modify (char *s1, char *s2) {
    char temp = *s1;
    *s1 = *s2;
    *s2 = temp;
    char *p1 = s1 + strlen(s1) - 1;
    *p1 = toupper(*p1);
    char *p2 = s2 + 1;
    *p2 = tolower(*p2);
    char *p3 = s1 + 2;
    *p3 = '#';
}
int main() {
```


Scanned with CamScanner

ourdiu.com | Shared by Sumit Saha

```
    char str1[50] = "Apple";
    char str2[50] = "BANANA";
    modify (str1, str2);
    printf("str1 = %s\n", str1);
    printf("str2 = %s\n", str2);
    printf("Sum of lengths = %d\n",
strlen(str1) + strlen(str2));
    return 0;
}
```

c) **03**

```
#include <stdio.h>
struct Student {
    char name[20];
    int marks;
};
void bonus(struct Student *s) {
    if (s->marks >= 50)
        s->marks += 5;
    else
        s->marks += 2;
}
int main() {
    struct Student s[3] = {{"Ali", 45}, {"Sara", 52}};
    for(int i = 0; i < 2; i++)
        bonus(&s[i]);
    for(int i = 0; i < 2; i++)
        printf("%s %d\n", s[i].name, s[i].marks);
    printf("Total marks = %d\n", s[0].marks + s[1].marks);
    printf("Average marks = %.2f\n", (s[0].marks + s[1].marks)/3.0);
    return 0;
}
```

3. **Identify** the problem scenarios given below and write a complete **C program** for each problem statement. **CO4**

a) **05**

During the DIU CSE Lab Recruitment Drive, each student submits an *interest tag* (a single word) that describes their primary area of interest (e.g., "coding", "robotics", "ai"). You must check whether the tag contains any digits. If it contains even one digit, the tag is considered invalid.

Write a C program that reads the tag and prints whether it is valid or not.

Input:

- One line containing the interest tag (a string without spaces)

Output:

- Print "**Valid Tag**" if the tag contains **no digits**
- Print "**Invalid Tag**" if **any digit** is found

Sample Input	Sample Output
coding32	Invalid Tag

Sample Input	Sample Output
kamonachan	Valid Tag

b) **05**

Each applicant rates themselves in 5 skill categories on a scale from 1 to 10. You are given an array of integers representing these skill ratings. Write a **recursive function** in C to print the entire array in reverse order.

Input:

- First line contains an integer n representing the number of skill ratings
- Second line contains n space-separated integers

Scanned with CamScanner

ourdiu.com | Shared by Sumit Saha

Output:

- Print the ratings in reverse order (space-separated)

Sample Input	Sample Output
5 3 7 8 6 9	9 6 8 7 3

c) A lab conducted a screening quiz where each student was evaluated in **three** categories: **05**

- **Problem Solving**
- **Logic**
- **Debugging**

The scores of all students are stored as a **2D** array of integers (each row represents a student). A student qualifies if:

- **Problem Solving** ≥ 7 AND (**Logic** ≥ 6 OR **Debugging** ≥ 6)

Write a C program that prints the score table and then prints the indices of all qualified students.

Input:

- First line contains an integer **n** (number of students)
- Next **n** lines each contain three integers: Problem Solving, Logic, Debugging scores

Output:

- Print the score table (one row per student)
- Print: "**Qualified: index**" for each qualified student

Sample Input	Sample Output
3 7 5 6-P 8 7 4-L 6 6 6-D	7 5 6 8 7 4 6 6 6 Qualified: 0 Qualified: 1

d) As part of a creativity task, students submit a codename (a single word). Write a user-defined function that receives this codename and returns: **05**

- 1 if the codename is a palindrome
- 0 otherwise

Use the function to determine and print whether the codename is a **palindrome**. A **palindrome** is any sequence of characters that reads the same forward and backward (e.g., noon, 1001). Palindromes can be words, phrases, numbers, or names.

Input:

- One line containing a codename (a string without spaces)

Output:

- Print "**Palindrome**" or "**Not Palindrome**"

Sample Input	Sample Output
level	Palindrome

Sample Input	Sample Output
label	Not Palindrome

Scanned with CamScanner

ourdiu.com | Shared by Sumit Saha

e) Define a structure Applicant with the following fields: 05

- **ID** (int)
- **Name** (string without spaces)
- **Event** (string without spaces)
- **Cleared** (1 = cleared screening, 0 = not cleared)

Write a C program to input details of n applicants and store them in an array of structures. Print the details of applicants who: have **cleared = 1** and belong to the "**coding**" event.

Input:

- First line contains integer n
- Next n blocks each contain:
  - Applicant ID
  - Name
  - Event
  - Cleared (1 or 0)

Output:

- Print details in this format:

  ID: <ID> Name: <Name> Event: <Event> Status: Cleared

Sample Input	Sample Output
3 101 Rafi coding 1 102 Sadia robotics 1 103 Imran coding 0	ID: 101 Name: Rafi Event: coding Status: Cleared

Scanned with CamScanner