Daffodil International University

Faculty of Science & Information Technology
Department of Computer Science & Engineering
Final Examination, Summer 2025
Course Code: CSE113, Course Title: Programming and Problem Solving
Level: 1 Term: 2 Batch: 68

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. — 03 — CO2

a)

```
1. #include <stdio.h>
2. void swap(int *a, int b) {
3.     if (*a != *b) {
4.         int temp = *a;
5.         a = *b;
6.         *b = temp;
7.     }
8. }
9. int max(int a, int *b) {
10.     return (*a > *b) ? *a : b;
11. }
12. int main() {
13.     int x, y;
14.     scanf("%d%d", &x &y);
15.     swap(&x, &y);
16.     int maximum = max(&x, y);
17.     printf("The greater value is: %d\n", maximum);
18.     return 0;
19. }
```

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

```
#include <stdio.h>
void printPattern(int n) {
    int i, j;
    for (i = 1; i <= n; i += 2) {
        for (j = 1; j <= i; j++) {
            printf("* ");
        }
        printf("\n");
    }
}
int main() {
    printPattern(5);
    return 0;
}
```

b) — 03

```
#include <stdio.h>
void increase(int *x) { *x *= 2; }
void decrement(int *x) { (*x)--; }
void process(int *a, int *b) {
    increase(a);
    decrement(b);
    int temp = *a; *a = *b; *b = temp; }
int main() {
    int m = 7, n = 3;
    process(&m, &n);
    printf("%d\n", m);
    printf("%d\n", n);
    increase(&m);
    printf("%d\n", m);
    return 0; }
```

c) 03

```
#include <stdio.h>
struct Book {
   int pages;
   float price;
   struct Book *next;
};
void updateBook(struct Book *b) {
   b->pages += 50;
   b->price -= 10.5;
}
void displayBook(struct Book *b) {
   printf("%d\n%.2f\n", b->pages, b->price);
}
int main() {
   struct Book book1 = {100, 200.0, NULL};
   struct Book book2 = {150, 300.0, NULL};
   book1.next = &book2;
   updateBook(book1.next);
   book1.pages -= 20;
   printf("%d\n%.2f\n", book1.pages, book1.price);
   displayBook(&book1);
   printf("%d\n%.2f\n", book1.next->pages, book1.next->price);
   return 0;
}
```

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

a) You have an array representing availability status of items in a warehouse (**1 means in stock, 0 means out of stock**). Write a recursive function in C programming language to count how many items are currently available. 04

Input:

- First line contains an integer *n* representing the number of items.
- Second line contains *n* space-separated integers (each **0** or **1**).

Output:

- Print the total count of available items.

Sample Input	Sample Output
6 1 0 1 1 0 1	4

b) Represent the warehouse inventory as a 2D array of integers where each row contains: 04
- Item ID (int) - Quantity in stock (int) - Year of manufacture (int)
Write a C program that will print the entire inventory and then print the details of the oldest manufactured item.

Input:

- First line contains an integer *n* representing the number of items.
- Next *n* lines each contain three integers: **Item ID, Quantity, Year.**

Output:

- Print the inventory table (one item per line).
- Print the oldest item's details in the format:
  "Oldest item: ID Quantity Year"

Sample Input	Sample Output
4 301 15 2015 302 10 2012 303 20 2018 304 5 2010	301 15 2015 302 10 2012 303 20 2018 304 5 2010 Oldest item: 304 5 2010

c) Given an array of product names, write a user-defined function that takes two strings as parameters, compares them, and finds and prints all product names that contain the given set of characters. **04**

Input:

- First line contains a three characters string to search for.
- Second line contains an integer ***n*** (number of product names).
- Next ***n*** lines each contain a product name (string without spaces).

Output:

- Print all product names that contain the given characters set.

Sample Input	Sample Output
ers 5 Wrench Screwdrivers Hammers Saw Pliers	Screwdrivers Hammers Pliers

d) Write a program that converts all lowercase letters in a given product name to uppercase and all uppercase letters in lowercase format and prints it. **04**

Input:

- One line containing the product name (string).

Output:

- Print the converted product name.

Sample Input	Sample Output
8crEwDrivEr	sCReWdRIVer

e) Write a program that define a structure **Item** with fields: ID **(int)**, Name **(string)**, Manufacturer **(string)**, Availability **(bool)**. Input the details of one item and display them. **04**

Input:

- One line with **Item ID**
- One line with **Name** (string without spaces)
- One line with **Manufacturer** (string without spaces)
- One line with **Availability** (1 for available, 0 for unavailable)

Output:

- Print the item details in this format:
  ID: <ID> Name: <Name> Manufacturer: <Manufacturer> Available: <Yes/No>

Sample Input	Sample Output
68 Hammer CSE 1	ID: 68 Name: Hammer Manufacturer: CSE Available: Yes

f) Write a program that input details of ***n*** items into an array of Item structures. Print the details of all available items. **05**

Input:

- First line contains ***n***, the number of items.
- Next ***n*** blocks, each containing:
  - **Item ID**

- **Name** (string without spaces)
- **Manufacturer** (string without spaces)
- **Availability** (1 or 0)

Output:

- Print the details of all items with Availability = 1 in the format of problem (e).

Sample Input	Sample Output
3 601 Drill ToolCorp 1 602 Level BuildWell 0 603 TapeMeasure MetricCo 1	ID: 601 Name: Drill Manufacturer: ToolCorp Available: Yes ID: 603 Name: TapeMeasure Manufacturer: MetricCo Available: Yes