

Final Exam

Fall 2023

Course Code: CS-1002	Course Name: Programming Fundamentals	
Instructor(s): Dr. M Shahzad, Dr. Farooque, Dr. Abdul Aziz, Zain, Basit, Sobia, Farooq Zaidi, M Kariz		
Date: 18th Dec 2023	Total Time: 3 hours, 09:00am – 12:00pm	Total Marks: 100
Student ID:	Section	

INSTRUCTIONS:

- Return the question paper and make sure to keep it inside your answer sheet.
- Read each question completely before answering it. There are 6 questions and 4 pages (two sided).
- In case of any ambiguity, you may make assumptions. However, your assumption should not contradict any statement in the question paper.
- Do not write anything on the question paper (except your ID and section).

QUESTION 1:[CLO: 1, TIME: 20 MINS, POINTS: 15]

Write on the answer sheet the output of the following programs, when they are executed. There are no compilation errors in the programs.

Part A.

```
#include <stdio.h>
struct Element {
    int value;
};
void recurse0p(struct Element arr[][3],
               int rows, int cols, int i, int j){
    if(i<rows){
        if(j < cols){
            printf("%d ",arr[i][j].value);
            recurse0p(arr, rows, cols, i,j+1);
        }
        else{
            printf("\n");
            recurse0p(arr, rows, cols, i+1,0);
        }
    }
}
int main() {
    struct Element arr[2][3] = {
        { {1}, {2}, {3} },
        { {4}, {5}, {6} }
    };
    recurse0p(arr, 2, 3, 0, 0);
    return 0;
} // end main
```

Part B.

```
#include <stdio.h>
int main() { int i,j;
    int arr1[] = {1, 2, 3};
    int arr2[] = {4, 5, 6};
    int arr3[] = {7, 8, 9};
    int *ptrArr[] = {arr1,arr2,arr3};
    printf("Original Array: \n");
    for ( i = 0; i < 3; ++i) {
        for ( j = 0; j < 3; ++j)
            printf("%d ", ptrArr[i][j]);
        printf("\n");
    } //end for i

    for ( i = 0; i < 3; ++i) {
        int *start = ptrArr[i];
        int *end = ptrArr[i] + 2;
        while (start < end) {
            int temp = *start;
            *start = *end;
            *end = temp;
            ++start; --end; } //end while
    } //end for i

    printf("Modified: \n");
    for ( i = 0; i < 3; ++i) {
        for ( j = 0; j < 3; ++j)
            printf("%d ", ptrArr[i][j]);
        printf("\n");
    } //end for i
    return 0;
} //end main
```

Part C.

```
#include<stdio.h>
int main(){   int i, j, k, n;
    n=7; //number of lines to be printed
    for (i=0; i<n; i++) {
        for(j=0; j<=n-i; j++)
            printf(" ");

        for(k=0; k<=i; k++)
            printf("%c ", 64+k+1);

        for(j=i; j>0; j--)
            printf("%c ", 64+j);

        printf("\n");
    } //end for i
    return 0; } //end main
```

QUESTION 2:[CLO: 2, TIME: 30 MINS, POINTS: 18 (6 EACH)]

Considering the output given, complete the following code snippets. [Attempt on answer script]

Part A.

```
#include <stdio.h>
struct student{
    -----
    -----
};
void writeStudentToFile(const char *filename) {
    -----
    -----
}
int main() {
    writeStudentToFile("student.txt");
    return 0;
}
```

OUTPUT:

Enter "exit" as First Name
to stop reading user input.

First Name: Ali
Last Name : Iqbal
Roll Number : 101
Percentage : 90.50

First Name: Naima
Last Name : Ali
Roll Number : 102
Percentage : 95.50

First Name: exit

Part B.

```
#include <stdio.h>
#define MAX_SIZE 5
int* getMinMax(int *array, const int size);
int main() {
    int array[MAX_SIZE] = {1, -2, 3, -1, 9};
    int *resultArr =getMinMax(array, MAX_SIZE);
    printf("Min value in array: %d\n", resultArr[0]);
    printf("Max value in array: %d\n", resultArr[1]);
    free(resultArr);
    return 0;}
int* getMinMax(int *numbers, const int size) {
    -----
    -----
}
```

Output:

Enter size of array: 5
Enter 5 elements in array:
1 -2 3 -1 9
Minimum value in array : -2
Maximum value in array : 9

Part C.

```
#include <stdio.h>
#include <string.h>
void removeWordFromString(char str[],
                          char word[], char neww[]) {
    -----
    -----
}
int main(){ char str[100], neww[100], word[100];
    printf("Enter string to remove a word from:");
    gets(str);
    printf("\nEnter the word you want removed: ");
    gets(word);
    removeWordFromString(str, word, neww);
    printf("\nAfter word removed: %s\n", neww);
    return 0; } //end main
```

Output:

Enter string to remove a word from: Programming Fundamental

Enter the word you want removed: gram

After word removed: Proming Fundamental

QUESTION 3:[CLO: 3, TIME: 25 MINS, POINTS: 12]

Consider a coinage system consisting of n coins. Each coin has a positive integer value. Your task is to produce a sum of money x using the available coins in such a way that the number of coins is minimal. For example, if the coins are $\{1, 5, 7\}$ and the desired sum is 11, an optimal solution is $5+5+1$ which requires 3 coins.

Write a recursive function `int foo(...)`, that returns the minimal number of coins to make the sum x .

You may assume that the input variables $\{arr, n, x\}$ are globally defined. You may write the function definition of `foo` with the parameters that you think are appropriate.

SAMPLE INPUT 1:

```
n = 3
x = 11
arr[n] = {1, 5, 7}
```

SAMPLE OUTPUT 1:

3

SAMPLE INPUT 2:

```
n = 4
x = 26
arr[n] = {2, 4, 8, 9}
```

SAMPLE OUTPUT 2:

4

QUESTION 4:[CLO: 2, TIME: 30 MINS, PIONTS: 20]

You need to write two functions for user authentication with encryption in C Language:

Part A. void encrypt(*usernames, *passwords): This function takes two pointer arrays as arguments: `usernames`: An array of 100 strings containing user names, and `passwords`: An array of 100 strings containing passwords. Strings are null (`'\0'`) terminated.

For each username and password pair, the function encrypts them using the below method:

- Each character in the string is replaced by another character that is i positions ahead in the alphabet.
- i is determined by the index of the string in the `usernames` array (e.g., first string element uses $i=0$, second element uses $i=1$, etc.).

Part B. int find(*usernames, *passwords, *search_username, *search_password): This function takes four arguments. The function searches in the encrypted `usernames` and `passwords` arrays for a matching pair corresponding to the provided `search_username` and `search_password` (un-encrypted). Function **returns 1** if a matching username and password pair are found, **0 otherwise**.

QUESTION 5:[CLO: 2, TIME: 35 MINS, POINTS: 15]

Develop a “C program” to manage electricity consumption and billing information. You are required to utilize structures to encapsulate details about the electricity usage for different days and implement a tiered billing system with different rates for various consumption levels. The program should analyze and display comprehensive information about a customer's electricity consumption.

Part A. Structures Definition [1+1+1 = 3 points]

- Structure “**DailyConsumption**” with members **day** representing the day of the month, **unitsConsumed** representing units of electricity consumed on that day.
- Structure “**ElectricityBill**” with members **customerName** representing the name of the customer, **customerID** representing the customer's unique identifier, and **dailyConsumptions** containing details for each day for 30 days.
- Structure “**BillingTier**” with members **rate** representing the rate per unit for the tier, **upperLimit** representing the upper limit for the tier(example given below). If -1.0, it indicates an unlimited upper limit.

```
struct BillingTier billingTiers[] = {  
    {0.10, 50.0},    // Rate for the first 50 units  
    {0.15, 100.0},   // Rate for the next 50 units  
    {0.20, -1.0}     // Rate for any units beyond 100 (unlimited)  
};
```

Part B. Functions [4+4+4 = 12 points]

1. Write a function “**calculateTotalConsumption**” to calculate and return the total units of electricity consumed. The function should print the total bill based on defined billing tiers as well.
2. Write a function “**findUnitFrequency**”, the function prints frequency of each day's units consumed. E.g. 200 units consumed on day 1, day 2 and day 7 then 200 units frequency is 3.
3. Write a function “**Analysis**” to display the days with the second highest and third lowest electricity consumption.

QUESTION 6:[CLO: 4, TIME: 30 MINS, POINTS: 20]

Implement a “C program” that dynamically allocates memory for strings and concatenates them. The program should perform the following steps:

1. **Input:**
 - Prompt the user to enter two strings of varying lengths.
 - Use dynamic memory allocation to create char arrays to store the input strings.
2. **Functionality:**
 - Create a function that takes the two input strings and dynamically allocates memory to concatenate them into a new string.
 - The concatenated string should have sufficient space for the combined strings and the null-terminator.
 - Repetitively take user inputs and concatenate until the users stops it with ‘Q’.
 - All new inputs must be concatenated with the previous data. Do not over-write previous data in the variables. (Hint: Something related to **re-allocation** might help)
3. **Output:** Display the original input strings and the concatenated result.
4. **Error Handling:** Implement appropriate error handling. Check for memory allocation failures and inform the user if there's an issue.
5. **Testing:** Test your program with strings of different lengths to ensure correct memory allocation, concatenation, and freeing of memory.