Ramakrishna Mission Vidyamandira

(An Autonomous College Under University of Calcutta)

Computer Science (Honors) Semester I 2023

Paper: 1CMSCOC1 Practical

Submitted by
Class Roll Number: 261 *Registration Number:* *B.Sc.* *1st Semester* *Batch: 2023-24*

INDEX

15	Write a program to get the summation of all the numbers in an array.	4/09/2023	11/09/2023	
16	Write a program to find out the duplicate elements in an array.	11/09/2023	18/09/2023	
17	Write a program to find out an element in an array.	11/09/2023	18/09/2023	
18	Write a program to sort elements of an array in ascending and descending order.	11/09/2023	18/09/2023	
19	Write a program to count the frequency of each element in an array.	18/09/2023	9/10/2023	
20	Write a menu-driven program to implement matrix-addition, subtraction and multiplication.	18/09/2023	9/10/2023	
21	Write a program to create a student database of N number of records.	18/09/2023	9/10/2023	
22	Write a program to check a string (case sensitive) is palindrome or not.	9/10/2023	16/10/2023	
23	Write a program to sort a list of strings in ascending and descending order.	9/10/2023	16/10/2023	
24	Write a program to read content of a ".txt" file and copy the content of the said file into another ".txt" file.	9/10/2023	16/10/2023	

Assignment No. 1

Problem Statement: Write a program to find out the G.C.D and L.C.M of two numbers.

Source code:

```c
#include<stdio.h>
int main()
{
    int a,b,t,x,y,z,LCM;
    printf("enter two numbers(a b): ");
    scanf("%d %d",&a,&b);
    if (a<0 || b<0)
    {
        printf("gcd and lcm of negative value is invalid");
        goto end;
    }
    x = a,y = b;
    while(a!=0)
    {
        t = a;
        a = b%a;
        b = t;
    }
    printf("value of G.C.D is %d\n",b);
    z = x*y;
    LCM = z/b;
    printf("value of L.C.M is %d\n",LCM);
    end:
        return 0;
    return 0;
}
```

Output:

Case 1:

```
enter two numbers(a b): 15 17
value of G.C.D is 1
value of L.C.M is 255
```

Case 2:

```
enter two numbers(a b): 12 14
value of G.C.D is 2
value of L.C.M is 84
```

Case 3:

```
enter two numbers(a b): -13 2
gcd and lcm of negative value is invalid
```

Conclusion:

In this C code , it calculates the Greatest Common Divisor (GCD) and the Least Common Multiple (LCM) of two input numbers. The user is prompted to input two integers, and the program proceeds to find their GCD through a while loop. Once the GCD is determined, the LCM is calculated using the relationship between GCD and LCM. The final results, GCD and LCM, are then displayed. There is not one solution to this problem. It could have been solved by other methods it can be improved by doing some changes.

Assignment No. 2

Problem Statement: Write a program to reverse a number.

Source code:

```c
#include<stdio.h>
int main()
{
    int a,sum=0;
    int rem;
    printf("enter the number to reverse: ");
    scanf("%d",&a);
    while(a != 0)
    {
        rem = a%10;
        sum = (10*sum)+rem;
        a = (int)a/10;
    }
    printf("reversed number is: %d",sum);
    return 0;
}
```

Output:

```
enter the number to reverse: 54275
reversed number is: 57245

enter the number to reverse: 9420579
reversed number is: 9750249

enter the number to reverse: -413434234
reversed number is: -432434314
```

Conclusion:

The provided C code is designed to reverse a given integer. The program prompts the user to input a number, and then it uses a while loop to iteratively extract the last digit of the number, store it in a new reversed number, and remove the last digit from the original number. The process continues until the original number becomes zero. Finally, the reversed number is displayed as the output. The code appears to be correctly implemented for reversing an integer, demonstrating a common technique using a while loop and simple arithmetic operations. There is not one solution to this problem. It could have been solved by other methods it can be improved by doing some changes.

Assignment No. 3

Problem Statement: Write a program to find out the the area of a triangle, square and rectangle.

Source code:

```c
#include<stdio.h>
#include<math.h>
#define AND &&
#define OR ||
float area_tri(float a,float b,float c);
int main()
{
    int choice;
    printf("enter which area do you want to get: \n");
    printf("1 for rectangle,2 for square,3 for triangle: ");
    scanf("%d",&choice);

    if (choice == 1)
    {
        //rectangle
        float length,breadth;
        printf("enter the value of length: ");
        scanf("%f",&length);
        printf("enter the value of breadth: ");
        scanf("%f",&breadth);
        printf("the area of the rectangle is: %f",length*breadth);
    }
    else if(choice == 2)
    {
        float side;
        printf("enter the value of one side: ");
        scanf("%f",&side);
        printf("the area of the square is: %f",side*side);
    }
    else if(choice == 3)
    {
        float a,b,c;
        printf("enter the value of three sides of the triangle:\n");
        printf("a: ");
        scanf("%f",&a);
        printf("b: ");
        scanf("%f",&b);
        printf("c: ");
        scanf("%f",&c);
        if (a+b>c AND a+c>b AND c+b>a)
            printf("the area of the triangle is %f",area_tri(a,b,c));
        else
            printf("triangle can not be created");
    }
    else
        printf("invalid input");
}

float area_tri(float a,float b,float c)
{
    float s,area;
    s = (a+b+c)/2;
    area = sqrt(s*(s-a)*(s-b)*(s-c));
    return area;
}
```

Output:

Case 1:

```
enter which area do you want to get:
1 for rectangle,2 for square,3 for triangle: 1
enter the value of length: 5.45
enter the value of breadth: 3.25
the area of the rectangle is: 17.712499
```

case 2:

```
enter which area do you want to get:
1 for rectangle,2 for square,3 for triangle: 2
enter the value of one side: 3.54
the area of the square is: 12.531600
```

case 3:

```
enter which area do you want to get:
1 for rectangle,2 for square,3 for triangle: 3
enter the value of three sides of the triangle:
a: 6
b: 4
c: 5
the area of the triangle is 9.921567
```

case 4:

```
enter which area do you want to get:
1 for rectangle,2 for square,3 for triangle: 3
enter the value of three sides of the triangle:
a: 6
b: 4
c: 15
triangle can not be created
```

Conclusion:

The provided C code is a simple program designed to calculate the area of a geometric shape based on user input. The user is prompted to choose the type of area they want to calculate—either for a rectangle, a square, or a triangle. The program then takes the necessary input values, performs the corresponding calculations, and prints the result.

The code uses conditional statements to handle different cases for rectangle, square, and triangle. For the triangle, there is an additional check to ensure the input values can form a valid triangle using the triangle inequality theorem. If the conditions are met, the program calls a separate function to calculate the area of the triangle using Heron's formula.

Enhancements can be made to the current solution by exploring alternative methods for this program.

Assignment No. 4

Problem Statement: Write a program to generate the following patterns for N number of rows:

i)

```
*
* *
* * *
* * * *
```

Source code:

```c
#include<stdio.h>
int main()
{
    int row,i,j;
    printf("enter a number: ");
    scanf("%d",&row);
    for(i=1;i<=row;i++)
    {
        for(j=1;j<=i;j++)
        {
            printf("* ");
            if(j == i)
            {
                printf("\n");
            }
        }
    }
}
```

Output:

```
enter a number: 4
*
* *
* * *
* * * *
```

Conclusion:

In conclusion, the provided C code generates a pattern of asterisks in the form of right-angled triangles based on user input. The user is prompted to enter a number, representing the number of rows in the pattern. The code then uses nested loops to iterate through each row and column, printing asterisks in a triangular pattern. The inner loop prints an asterisk and checks if the

column number is equal to the row number, indicating the end of a row. If true, it adds a newline character to move to the next row. The solution to this program is not exclusive. Introducing modifications could lead to better outcomes.

ii)

```
      *
     **
    ***
   ****
```

Source code:

```c
#include<stdio.h>
int main()
{
    int a,j;
    printf("enter the number: ");
    scanf("%d",&a);
    for(int i=1;i<=a;i++)
    {
        for(j=1;j<=a-i;j++)
        {
            printf(" ");
        }
        for(int k=j;k<=a;k++)
        {
            printf("*");
        }
        printf("\n");
    }
}
```

Output:

```
enter the number: 4
   *
  **
 ***
****
```

Conclusion:

In conclusion, the provided C code generates a pattern of asterisks in the form of a right-angled triangle with the right-angle on the left side. The user is prompted to enter a number, which determines the height of the triangle. The code then utilizes a combination of nested loops to achieve the pattern. The outer loop controls the rows, the first inner loop prints leading spaces, and the second inner loop prints the asterisks. The number of asterisks in each row is determined

by the current value of 'a' and the loop variables.The solution to this program is not exclusive; introducing modifications could lead to better outcomes.

iii)

```
          *
        *   *
      *   *   *
    *   *   *   *
```

Source code:

```c
#include<stdio.h>
int main()
{
    int a,j;
    printf("enter the number: ");
    scanf("%d",&a);
    for(int i=1;i<=a;i++)
    {
        for(j=1;j<=a-i;j++)
        {
            printf(" ");
        }
        for(int k=j;k<=a;k++)
        {
            printf("* ");
        }
        printf("\n");

    }
}
```

Output:

```
enter the number: 4
      *
    *   *
  *   *   *
*   *   *   *
```

Conclusion:

In conclusion, the provided C code creates a pattern of asterisks in the shape of a triangle. The user is prompted to input a number, determining the height of the triangle. The code employs nested loops to control the row and column printing. The first inner loop handles the leading spaces based on the row number, while the second inner loop prints asterisks followed by a space.The solution to this program is not exclusive; introducing modifications could lead to better outcomes.

Assignment No. 5

Problem Statement: Write a program to convert temperature from Centigrade to Fahrenheit and vice-versa.

Source code:

```c
#include<stdio.h>
float c_to_f(float c);
float f_to_c(float f);
int main()
{
    int choice;
    printf("enter which do you want to do: \n");
    printf("1 for C to F, 2 for F to C: \n");
    scanf("%d",&choice);
    if (choice == 1)
    {
        float C;
        printf("enter temperature in C: ");
        scanf("%f",&C);
        printf("%f Celsius to Fahrenheit is %f",C,c_to_f(C));
    }
    else if(choice == 2)
    {
        float F;
        printf("enter temperature in F: ");
        scanf("%f",&F);
        printf("%f Fahrenheit to Celsius is %f",F,f_to_c(F));
    }
    return 0;
}

float c_to_f(float c)
{
    float f;
    f = ((9*c)/5)+32;
    return f;
}

float f_to_c(float f)
{
    float c;
    c = ((f-32)/9)*5;
    return c;
}
```

Output:

Case 1:

```
enter which do you want to do:
1 for C to F, 2 for F to C:
1
enter temperature in C: 100.34
100.339996 Celsius to Fahrenheit is 212.612000
```

Case 2:

```
enter which do you want to do:
1 for C to F, 2 for F to C:
2
enter temperature in F: 32.22
32.220001 Fahrenheit to Celsius is 0.122223
```

Conclusion:

In conclusion, the provided C code is a simple temperature conversion program that allows users to convert temperatures between Celsius and Fahrenheit. The user is prompted to choose the conversion type: from Celsius to Fahrenheit (choice 1) or from Fahrenheit to Celsius (choice 2). Depending on the choice, the program then prompts the user to input the temperature in the specified unit and calculates and displays the converted temperature using the respective conversion functions.

The code includes separate functions for converting Celsius to Fahrenheit (c_to_f) and Fahrenheit to Celsius (f_to_c).The solution to this program is not exclusive; introducing modifications could lead to better outcomes, such as considering alternative algorithms or refining the existing approach.

Assignment No. 6

Problem Statement: Write a program to generate a mark sheet based on “Gradation” for a given marks

Source code:

```c
#include<stdio.h>
#define AND &&
int main()
{
    int mark;
    printf("enter your obtained marks: ");
    scanf("%d",&mark);

    if(mark>=80 AND mark<=100)
        printf("your grade is AA");
    else if(mark<80 AND mark>=75)
        printf("your grade is A");
    else if(mark<75 AND mark>=70)
        printf("your grade is B");
    else if(mark<70 AND mark>=60)
        printf("your grade is C");
    else if(mark<60 AND mark>=0)
        printf("your grade is D");
    else if(mark>100)
        printf("Invalid input");
    else
        printf("Invalid input");
}
```

Output:

Case 1:

```
enter your obtained marks: 50
your grade is D
```

case 2:

```
enter your obtained marks: 60
your grade is C
```

case 3:

```
enter your obtained marks: 70
your grade is B
```

case 4:

```
enter your obtained marks: 80
your grade is AA
```

case 5:

```
enter your obtained marks: -2
Invalid input
```

case 6:

```
enter your obtained marks: 101
Invalid input
```

conclusion:

The provided C code is a simple program that determines the grade based on the user's obtained marks. The user is prompted to input their marks, and the program uses a series of if-else statements to evaluate the range of marks and assign the corresponding grade. The code defines a macro (AND) to represent the logical "and" operator for clarity.

The grading system is set up with different ranges for grades AA, A, B, C, and D. If the entered marks fall within a specific range, the program prints the corresponding grade. If the entered marks are out of the valid range (less than 0 or greater than 100), the program outputs an "Invalid input" message.

Optimal solutions to this program aren't rigid; incorporating changes might lead to better results.

Assignment No. 7

Problem Statement: Write a program to find out the prime numbers within a given range.

Source code:

```c
#define OR ||
#include<stdio.h>
#include<math.h>
int check_prime(int);
int main()
{
    int n,a;
    printf("enter the range like(a b): ");
    scanf("%d %d",&a,&n);
    for(a;a<=n;a++)
    {
        if (check_prime(a) == 0)
            printf("%d is a prime number\n",a);
        else if(check_prime(a) == 2)
            printf("%d is neither a prime nor non-prime number\n",a);
        else if(check_prime(a) == 3)
            printf("%d is a negative number\n",a);
    }
}
int check_prime(int n)
{
    if(n<0)
        return 3; // representing negative numbers
    else if ((n == 0) OR (n == 1))
        return 2; // neither prime nor non-prime
    else
    {
        int count = 0;
        int range = sqrt(n) + 1;
        for(int i = 1;i<range;i++)
        {
            if (n%i == 0)
            {
                count += 1;
                if (count > 1)
                {
                return 1; // for non-prime numbers
                break;
                }
            }
        }
        if (count == 1)
        {
            return 0; // for prime numbers
        }
    }
}
```

Output :

Case 1:

```
enter the range like(a b): 30 40
31 is a prime number
37 is a prime number
```

case 2:

```
enter the range like(a b): -2 0
-2 is a negative number
-1 is a negative number
0 is neither a prime nor non-prime number
```

conclusion:

The provided C code is designed to check and classify numbers within a specified range as prime, non-prime, or negative. The user is prompted to input a range (two integers), and the program then iterates through the numbers within that range. For each number, the check_prime function is called to check if the number is prime or not.

The check_prime function handles the classification logic. If a number is negative, it returns 3, representing a negative number. If a number is 0 or 1, it returns 2, indicating neither prime nor non-prime. For other positive numbers, the function checks for divisors up to the square root of the number to determine whether the number is prime or not.

The main function then prints the classification result based on the returned values from check_prime. The code effectively utilizes a macro (OR) to represent the logical "or" operator for improved readability.

This problem can also be improved using other algorithm, so, this is not only the exclusive solution.

Assignment No. 8

Problem Statement: Write a program to find out the factors of a given number.

Source code:

```c
#include<stdio.h>
int main()
{
    int n;
    printf("enter the number: ");
    scanf("%d",&n);
    printf("factors of the number %d is\n",n);
    for(int i = 1;i<(n+1);i++)
    {
        if(n%i == 0)
        {
            printf("%d ",i);
        }
    }
}
```

Output:

Case 1:

```
enter the number: 15
factors of the number 15 is
1 3 5 15
```

Case 2:

```
enter the number: 7
factors of the number 7 is
1 7
```

Case 3:

```
enter the number: 120
factors of the number 120 is
1 2 3 4 5 6 8 10 12 15 20 24 30 40 60 120
```

Conclusion:

The provided C program takes a user-input number and then calculates and prints its factors. It employs a simple for loop to iterate through potential factors, identifying and displaying each factor of the entered number.

Assignment No. 9

Problem Statement: Write a program to calculate the factorial of a given number.

Source code:

```c
#include<stdio.h>
int main()
{
    int n,fact = 1;
    printf("enter the number: ");
    scanf("%d",&n);
    if (n >= 0)
    {
        if (n == 0)
            fact = 1;
        else
        {
            for(int i = 1;i<=n;i++)
                fact *= i;
        }
        printf("factorial of %d is = %d",n,fact);
    }
    else if(n < 0)
        printf("not defined");
    return 0;

}
```

Output:

Case 1:

```
enter the number: 0
factorial of 0 is = 1
```

case 2:

```
enter the number: 5
factorial of 5 is = 120
```

case 3:

```
enter the number: -5
not defined
```

Conclusion:

The provided C code efficiently calculates and prints the factorial of a given non-negative integer. The program prompts the user to input a number (n), and based on the input, it uses a conditional structure to handle different scenarios. If n is greater than or equal to zero, the code employs a for loop to iteratively compute the factorial. In the case of n being zero, the factorial is set to 1. If n is negative, the program outputs "not defined" since factorials are not defined for negative numbers.There isn't a single solution to tackle this program; alternative methods could enhance its efficiency.

Assignment No. 10

Problem Statement: Write a program to generate the Fibonacci series up to Nth term.

Source code:

```c
#include<stdio.h>
int main()
{
    int n,a,b,i,z,t;
    a = 0;b = 1;
    printf("enter the n th number: ");
    scanf("%d",&n);

    for(i = 0;i<n;i++)
    {
        if((i==0)||(i==1))
        {
            printf("%d ",i);
        }
        else
        {
            z = a+b;
            t = b;
            b = z;
            a = t;
            printf("%d ",z);
        }
    }

    return 0;
}
```

Output:

```
enter the n th number: 6
0 1 1 2 3 5
```

Conclusion:

The provided C code generates and prints the Fibonacci series up to the nth term. The user is prompted to input the value of 'n,' and the program uses a for loop to iteratively calculate and display the Fibonacci sequence. The code initializes the first two terms (a and b) to 0 and 1, respectively.

The loop then follows a specific logic: for the first two terms, it prints the term itself. For subsequent terms, it calculates the next Fibonacci number (z) as the sum of the previous two

terms (a and b). The variables are then updated to prepare for the next iteration.

Assignment No. 11

Problem Statement: Write a menu-driven program to implement different arithmetic operations.

Source code:

```c
#include<stdio.h>
int main()
{
    int choice;
    printf("enter your choice: \n");
    printf("1 for addition\n");
    printf("2 subtraction\n");
    printf("3 multiplication\n");
    printf("4 division\n");
    scanf("%d",&choice);
    if (choice < 0 || choice > 4){
        printf("invalid input");
    }
    else{
       float a,b;
    printf("first num: ");
    scanf("%f",&a);
    printf("second num: ");
    scanf("%f",&b);
    if (choice == 1)
    {
        printf("%f + %f = %f",a,b,a+b);
    }
    else if(choice == 2)
    {
        printf("%f - %f = %f",a,b,a-b);
    }
    else if(choice == 3)
    {
        printf("%f x %f = %f",a,b,a*b);
    }
    else if(choice == 4)
    {
        if (b == 0)
            printf("division by Zero not possible!!");
        else
            printf("%f / %f = %f",a,b,a/b);
    }
    else
```

```
        printf("invalid input");
    }}
```

Output:

Case 1:

```
enter your choice:
1 for addition
2 substraction
3 multiplication
4 division
1
first num: 34
34.000000 + 23.000000 = 57.000000
```

case 2:

```
enter your choice:
1 for addition
2 substraction
3 multiplication
4 division
2
first num: 45
second num: 23
45.000000 - 23.000000 = 22.000000
```

case 3:

```
enter your choice:
1 for addition
2 substraction
3 multiplication
4 division
3
first num: 23
second num: 5
23.000000 x 5.000000 = 115.000000
```

case 4:

```
enter your choice:
1 for addition
2 substraction
3 multiplication
4 division
```

```
4
first num: 34
second num: 2
34.000000 / 2.000000 = 17.000000
```

case 5:

```
enter your choice:
1 for addition
2 substraction
3 multiplication
4 division
4
first num: 23
second num: 0
division by Zero not possible!!
```

Case 6:

```
enter your choice:
1 for addition
2 subtraction
3 multiplication
4 division
-1
invalid input
```

Conclusion:

The provided C code is a simple calculator program that performs basic arithmetic operations based on user input. It prompts the user to choose an operation (addition, subtraction, multiplication, or division), and then input two numbers. The code includes input validation to handle cases where the user enters an invalid operation choice. Additionally, it checks for division by zero to avoid potential errors.Enhancements can be made to the current solution by exploring alternative methods for this program.

Assignment No. 12

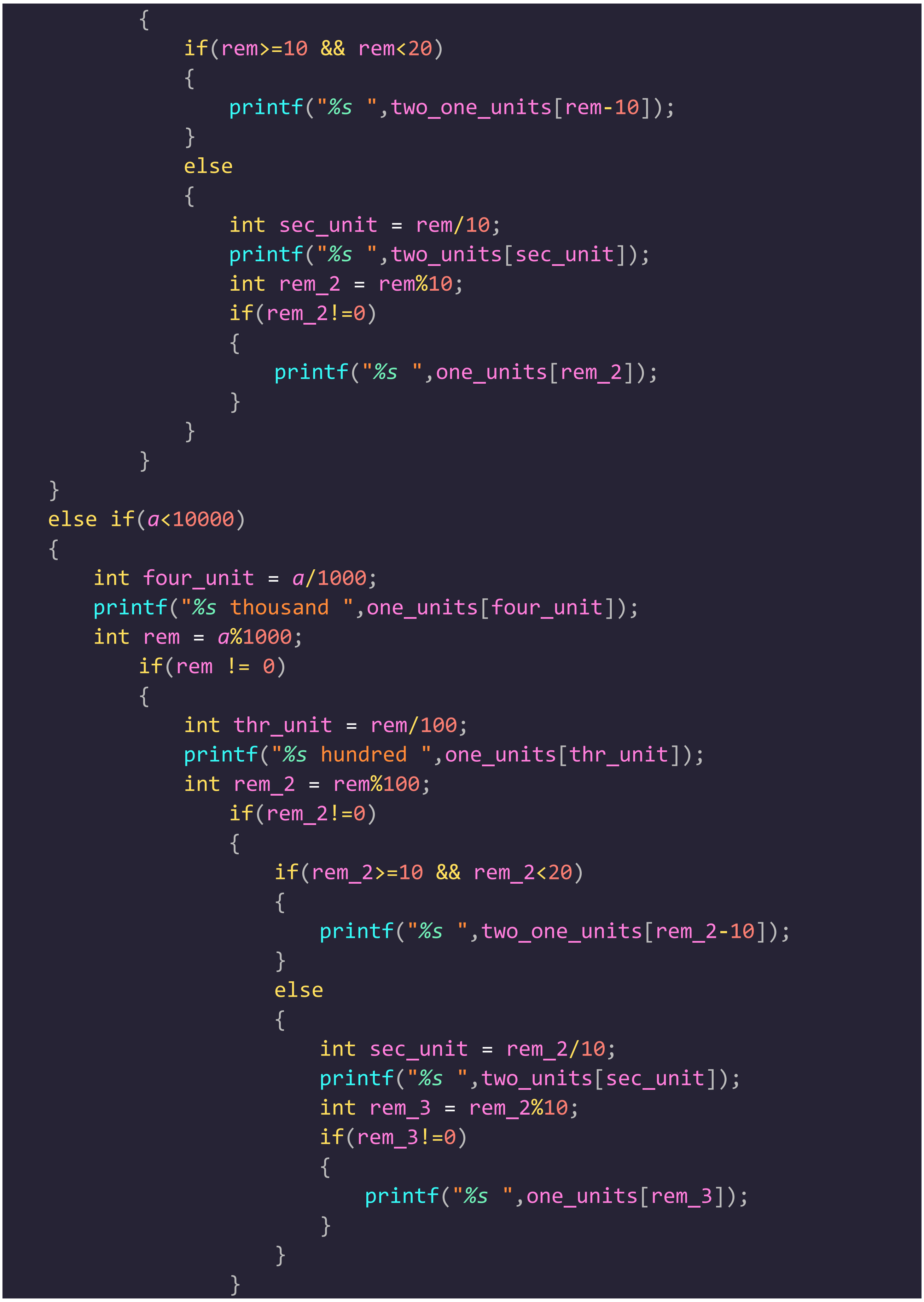
Problem Statement: Write a program to print a number in word.

Source code:

```
#include<stdio.h>
void num_to_word(int a)
{
    char *one_units[] = {"zero", "one", "two", "three", "four", "five", "six",
"seven", "eight", "nine"};


     char *two_one_units[] = {"ten", "eleven", "twelve", "thirteen",
"fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen"};


    char *two_units[] = {"", "", "twenty", "thirty", "forty", "fifty", "sixty",
"seventy", "eighty", "ninety"};
    if(a<10)
    {
        printf("%s",one_units[a]);
    }
    else if(a<100)
    {
        if(a>=10 && a<20)
        {
            printf("%s",two_one_units[a-10]);
        }
        else
        {
            int sec_unit = a/10;
            printf("%s ",two_units[sec_unit]);
            int rem = a%10;
                if(rem!=0)
                {
                    printf("%s",one_units[rem]);
                }
        }
    }
    else if(a<1000)
    {
        int thr_unit = a/100;
        printf("%s hundred ",one_units[thr_unit]);
        int rem = a%100;
            if(rem!=0)
```



```c
            {
                if(rem>=10 && rem<20)
                {
                    printf("%s ",two_one_units[rem-10]);
                }
                else
                {
                    int sec_unit = rem/10;
                    printf("%s ",two_units[sec_unit]);
                    int rem_2 = rem%10;
                    if(rem_2!=0)
                    {
                        printf("%s ",one_units[rem_2]);
                    }
                }
            }
    }
    else if(a<10000)
    {
        int four_unit = a/1000;
        printf("%s thousand ",one_units[four_unit]);
        int rem = a%1000;
            if(rem != 0)
            {
                int thr_unit = rem/100;
                printf("%s hundred ",one_units[thr_unit]);
                int rem_2 = rem%100;
                    if(rem_2!=0)
                    {
                        if(rem_2>=10 && rem_2<20)
                        {
                            printf("%s ",two_one_units[rem_2-10]);
                        }
                        else
                        {
                            int sec_unit = rem_2/10;
                            printf("%s ",two_units[sec_unit]);
                            int rem_3 = rem_2%10;
                            if(rem_3!=0)
                            {
                                printf("%s ",one_units[rem_3]);
                            }
                        }
                    }
            }
    }
    else if(a<100000)
    {
        int lst_unit = a/1000;
        if(lst_unit>=10 && lst_unit<20)
        {
            printf("%s",two_one_units[lst_unit-10]);
        }
        else
        {
            int sec_unit = lst_unit/10;
            printf("%s ",two_units[sec_unit]);
            int rem = lst_unit%10;
                if(rem!=0)
                {
                    printf("%s",one_units[rem]);
                }
        }
        printf(" thousand ");
        int rem = a%1000;
            if(rem != 0)
            {
                int thr_unit = rem/100;
                printf("%s hundred ",one_units[thr_unit]);
                int rem_2 = rem%100;
                    if(rem_2!=0)
                    {
                        if(rem_2>=10 && rem_2<20)
                        {
                            printf("%s ",two_one_units[rem_2-10]);
                        }
                        else
                        {
                            int sec_unit = rem_2/10;
                            printf("%s ",two_units[sec_unit]);
                            int rem_3 = rem_2%10;
                            if(rem_3!=0)
                            {
                                printf("%s ",one_units[rem_3]);
                            }
                        }
                    }
            }
    }
}
int main()
{
    int a;
    printf("enter a number between(-100000 - 100000): ");
    scanf("%d",&a);
    if(a<0)
    {
        printf("minus ");num_to_word(-a);
    }
    else
        num_to_word(a);
}
```

Output:

Case1:

```
enter a number between(-100000 - 100000): 41
forty one
```

Case2:

```
enter a number between(-100000 - 100000): -41
minus forty one
```

Conclusion:

The provided C program efficiently converts an input integer, ranging from -100000 to 100000, into its word representation. The program utilizes arrays to store words for single digits, teens, tens, hundreds, and thousands. It handles both positive and negative numbers, offering a clear and concise word representation for each digit in the input.

Assignment No. 13

Problem Statement:Write a program to check a character is an alphabet, digit or special character.

Source code:

```c
#include<stdio.h>
#define AND &&
#define OR ||
int main()
{
    char a;
    printf("enter the character: ");
    scanf("%c",&a);
    if((a >= 33 AND a<= 47) OR (a >= 58 AND a<= 64) OR (a >= 91 AND a<= 96)
OR (a >= 123 AND a<= 126))
        printf("%c is a special character\n",a);
    else
        if (a >= 48 AND a <= 57)
            printf("%c is a digit\n",a);
        else
            if((a >= 65 AND a <= 90) OR (a>=97 AND a<=122))
                printf("%c is an alphabet\n",a);
            else
                printf("Unknown Character");
    return 0;
}
```

Output:

Case 1:

```
enter the character: a
a is an alphabet
```

case 2:

```
enter the character: !
! is a special character
```

Case 3:

```
enter the character: 2
2 is a digit
```

Conclusion:

The provided C code is a character classification program that determines whether a given input

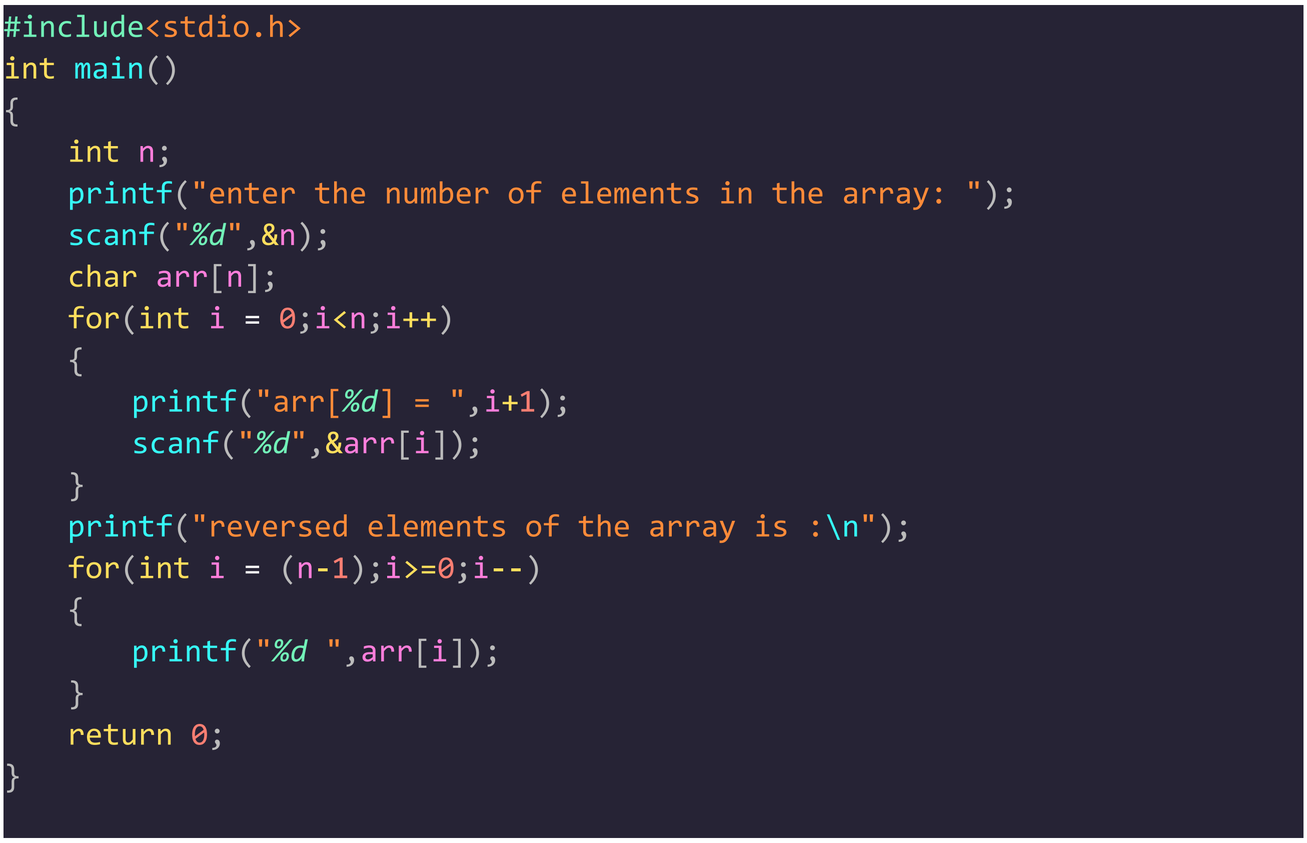
character falls into the categories of special characters, digits, alphabets, or an unknown character type. The user is prompted to enter a character, and the program uses a series of nested conditional statements to evaluate and classify the input character based on its ASCII value.

The code effectively employs the logical operators AND and OR, defined using the #define preprocessor directive, to create a comprehensive set of conditions for categorizing the character. It covers special characters, digits, and alphabets, providing clear output messages for each case.

Assignment No. 14

Problem Statement: Write a program to print the elements of an array in reverse order.

Source code:



```c
#include<stdio.h>
int main()
{
    int n;
    printf("enter the number of elements in the array: ");
    scanf("%d",&n);
    char arr[n];
    for(int i = 0;i<n;i++)
    {
        printf("arr[%d] = ",i+1);
        scanf("%d",&arr[i]);
    }
    printf("reversed elements of the array is :\n");
    for(int i = (n-1);i>=0;i--)
    {
        printf("%d ",arr[i]);
    }
    return 0;
}
```

Output:

```
enter the number of elements in the array: 5
arr[1] = 34
arr[2] = 2
arr[3] = 65
arr[4] = 87
arr[5] = 23
reversed elements of the array is :
23 87 65 2 34
```

Conclusion:

The provided C code is designed to reverse the elements of an array based on user input. The program prompts the user to enter the number of elements for the array, dynamically allocates memory for the array using variable-length array (VLA) syntax, and then initializes each array element with user-input values. After populating the array, the program prints the reversed order of the elements.

Assignment No. 15

Problem Statement: Write a program to get the summation of all the numbers in an array.

Source code:

```c
#include<stdio.h>
int main()
{
    int a;
    printf("enter the number of the elements in the array: ");
    scanf("%d",&a);
    char arr[a];
    for(int i = 0;i<a;i++)
    {
        printf("arr[%d] = ",i+1);
        scanf("%d",&arr[i]);
    }
    int sum = 0;
    for(int i = 0;i<a;i++)
    {
        sum += arr[i];
    }
    printf("total sum is %d",sum);
}
```

Output:

```
enter the number of the elements in the array: 10
arr[1] = 1
arr[2] = 2
arr[3] = 3
arr[4] = 4
arr[5] = 5
arr[6] = 6
arr[7] = 7
arr[8] = 8
arr[9] = 9
arr[10] = 10
total sum is 55
```

conclusion:

The provided C code is designed to calculate the sum of elements in an array. The user is prompted to input the number of elements for the array, and the program dynamically allocates memory for the array using variable-length array (VLA) syntax. The code then proceeds to collect

user input for each array element and calculates the sum of all elements using a for loop.This is not a unique solution to solve this program; it could have been solved by other methods.

Assignment No. 16

Problem Statement: Write a program to find out the duplicate elements in an array.

Source code:

```c
#include<stdio.h>
void check(int *i,int *j,int arr[],int *a,int *count);

int main()
{
    int count =1,a;
    printf("enter the number of elements in the array: ");
    scanf("%d",&a);
    int arr[a];

    for(int i = 0;i<a;i++)
    {
        printf("arr[%d] = ",i+1);
        scanf("%d",&arr[i]);
    }

    for (int i = 0;i<a;i++)
    {
        for(int j = 0 ;j<a;j++)
        {
            if(i!=j)
            {
                check(&i,&j,arr,&a,&count);
            }
        }
        if(count != 1)
        {
        printf("duplicate element is %d \n",arr[i]);
        }
        count = 1;
    }
}

void check(int *i,int *j,int arr[],int *a,int *count)
{
    if(arr[*i] == arr[*j])
    {
        *count += 1;
        int k;
        for(k = *j;k<(*a-1);k++)
        {
            arr[k] = arr[k+1];
        }
        *a -= 1;
        *j -= 1;
    }
}
```

Output:

Case 1:

```
enter the number of elements in the array: 4
arr[1] = 2
arr[2] = 3
arr[3] = 2
arr[4] = 5
duplicate element is 2
```

Case 2:

```
arr[1] = 2
arr[2] = 5
arr[3] = 2
arr[4] = 8
arr[5] = 6
arr[6] = 3
arr[7] = 2
arr[8] = 8
arr[9] = 3
arr[10] = 5
duplicate element is 2
duplicate element is 5
duplicate element is 8
duplicate element is 3
```

Conclusion:

The provided C code is designed to identify and display duplicate elements in an array. The user is prompted to input the number of elements for the array, and the program dynamically allocates memory for the array using variable-length array (VLA) syntax. The code then collects user input for each array element.

The main logic of the program involves nested loops to compare each element with every other element in the array. The check function is called to determine if there are duplicate elements, and if found, it adjusts the array to eliminate duplicates.

It can be improved by modification in the code.

Assignment No. 17

Problem Statement: Write a program to find out an element in an array.

Source code:

```c
#include<stdio.h>
int main()
{
    int a,n,count = 0;
    printf("enter the number of the elements in the array: ");
    scanf("%d",&a);
    char arr[a];
    for(int i = 0;i<a;i++)
    {
        printf("arr[%d] = ",i+1);
        scanf("%d",&arr[i]);
    }

    printf("enter the element to find: ");
    scanf("%d",&n);
    for(int i = 0;i<a;i++)
    {
        if (arr[i]==n)
        {
            count++;
            printf("the element is found!!\n");
            printf("index number is %d\n",i+1);
        }
    }
    if(count==0)
        printf("there is no existance of this element in the array!!");
}
```

Output:

Case 1:

```
enter the number of the elements in the array: 7
arr[1] = 1
arr[2] = 2
arr[3] = 3
arr[4] = 4
arr[5] = 5
arr[6] = 6
```

```
arr[7] = 7
enter the element to find: 3
the element is found!!
index number is 3
```

case 2:

```
enter the number of the elements in the array: 7
arr[1] = 2
arr[2] = 3
arr[3] = 6
arr[4] = 2
arr[5] = 7
arr[6] = 7
arr[7] = 4
enter the element to find: 9
there is no existance of this element in the array!!
```

conclusion:

The provided C code is a program that searches for a specific element in an array. The user is prompted to input the number of elements for the array, and the program dynamically allocates memory for the array using variable-length array (VLA) syntax. The code then collects user input for each array element.

Subsequently, the user is prompted to input an element to search for within the array. The program uses a for loop to iterate through the array and checks if the input element matches any of the elements in the array. If a match is found, it prints a message indicating the presence of the element, along with its index. If no match is found, it outputs a message indicating the absence of the element in the array.

Assignment No. 18

Problem Statement: Write a program to sort elements of an array in ascending and descending order.

Source code:

```c
#include<stdio.h>
void swap(int *a,int *b); // to swap to numbers
void printarr(int arr[],int a); // to print array elemnts
int length(int arr[],int size);
int main()
{
    int a,count,c;

    printf("number of elements in the array: ");

    scanf("%d",&a);
    int arr[a];
    for(int i = 0;i<a;i++)
    {
        printf("arr[%d] :",i+1);
        scanf("%d",&arr[i]);
    }

    printf("\n");

    printf("enter 1 for ascending, or 2 for descending:  ");
    scanf("%d",&c);

    if (c == 1)
    {
        do{
            count = 0;
            for(int i = 0;i<a-1;i++)
            {
                if(arr[i]>arr[i+1])
                {
                    swap(&arr[i],&arr[i+1]);
                    count++;
                }
            }
        }while(count != 0);
        printarr(arr,a);
    }
    else if(c == 2)
    {
        do{
            count = 0;
            for(int i = 0;i<a-1;i++)
            {
                if(arr[i]<arr[i+1])
                {
                    swap(&arr[i],&arr[i+1]);
                    count++;
                }
            }
        }while(count != 0);
        printarr(arr,a);
    }
    else
    {
        printf("wrong input!!");
    }
}
void swap(int *a,int *b)
{
    int t;
    t = *a;
    *a = *b;
    *b = t;
}
void printarr(int arr[],int a)
{
    printf("{");
    for(int i = 0;i<a;i++)
    {
        printf("%d,",arr[i]);
    }
    printf("}");
}
int length(int arr[],int size)
{
    return size;
}
```

Output:

Case 1:

```
number of elements in the array: 5
arr[1] :12
```

```
arr[2] :23
arr[3] :34
arr[4] :45
arr[5] :56

enter 1 for ascending, or 2 for descending:    1
{12,23,34,45,56,}


case 2:
number of elements in the array: 5
arr[1] :12
arr[2] :23
arr[3] :34
arr[4] :45
arr[5] :56

enter 1 for ascending, or 2 for descending:    2
{56,45,34,23,12,}
```

Conclusion:

The provided C program uses the Bubble Sort algorithm to arrange an array in ascending or descending order, determined by user input. The program initiates by prompting the user to specify the number of elements in the array and then proceeds to take input for each element. The sorting direction (ascending or descending) is also provided by the user. The implementation of the Bubble Sort algorithm is evident, where the array elements are iteratively compared and swapped until the entire array is sorted.

This is not a unique solution to solve this program; it could have been solved by other methods. It can be improved by some changes, such as considering alternative algorithms like merge sort , selection sort.

Assignment No. 19

Problem Statement: Write a program to count the frequency of each element in an array.

Source code:

```c
#include<stdio.h>
int len(int arr[],int size);
void check(int *i,int *j,int arr[],int *a,int *count);
int main()
{
    int count =1;
    int arr[]={67,98,67,67,34,34};
    int a = len(arr,sizeof(arr)/sizeof(arr[0]));

    for (int i = 0;i<a;i++)
    {
        for(int j = 0 ;j<a;j++)
        {
            if(i!=j)
            {
                check(&i,&j,arr,&a,&count);
            }
        }
        printf("%d present there %d times\n",arr[i],count);
        count = 1;
    }
}
void check(int *i,int *j,int arr[],int *a,int *count)
{
    if(arr[*i] == arr[*j])
    {
        *count += 1;
        int k;
        for(k = *j;k<(*a-1);k++)
        {
            arr[k] = arr[k+1];
        }
        *a -= 1;
        *j -= 1;
    }
}
int len(int arr[],int size)
{
    return size;
}
```

Output:

Case 1:

```
number of elements in the array: 5
arr[1] :1
arr[2] :1
arr[3] :1
arr[4] :1
arr[5] :1
1 present there 5 times
```

Case 2:

```
number of elements in the array: 5
arr[1] :1
arr[2] :2
arr[3] :1
arr[4] :1
arr[5] :2
1 present there 3 times
2 present there 2 times
```

Case 3:

```
number of elements in the array: 5
arr[1] :1
arr[2] :2
arr[3] :3
arr[4] :4
arr[5] :5
1 present there 1 times
2 present there 1 times
3 present there 1 times
4 present there 1 times
5 present there 1 times
```

Conclusion:

The provided C program effectively counts and displays the frequency of each element in an array using a nested loop and a separate function for updating counts and eliminating duplicates. The program dynamically adjusts the array size to reflect the removal of duplicates.

This is not a unique solution to solve this program; it could have been solved by other methods. It can be improved by some changes, such as considering alternative algorithms.

Assignment No. 20

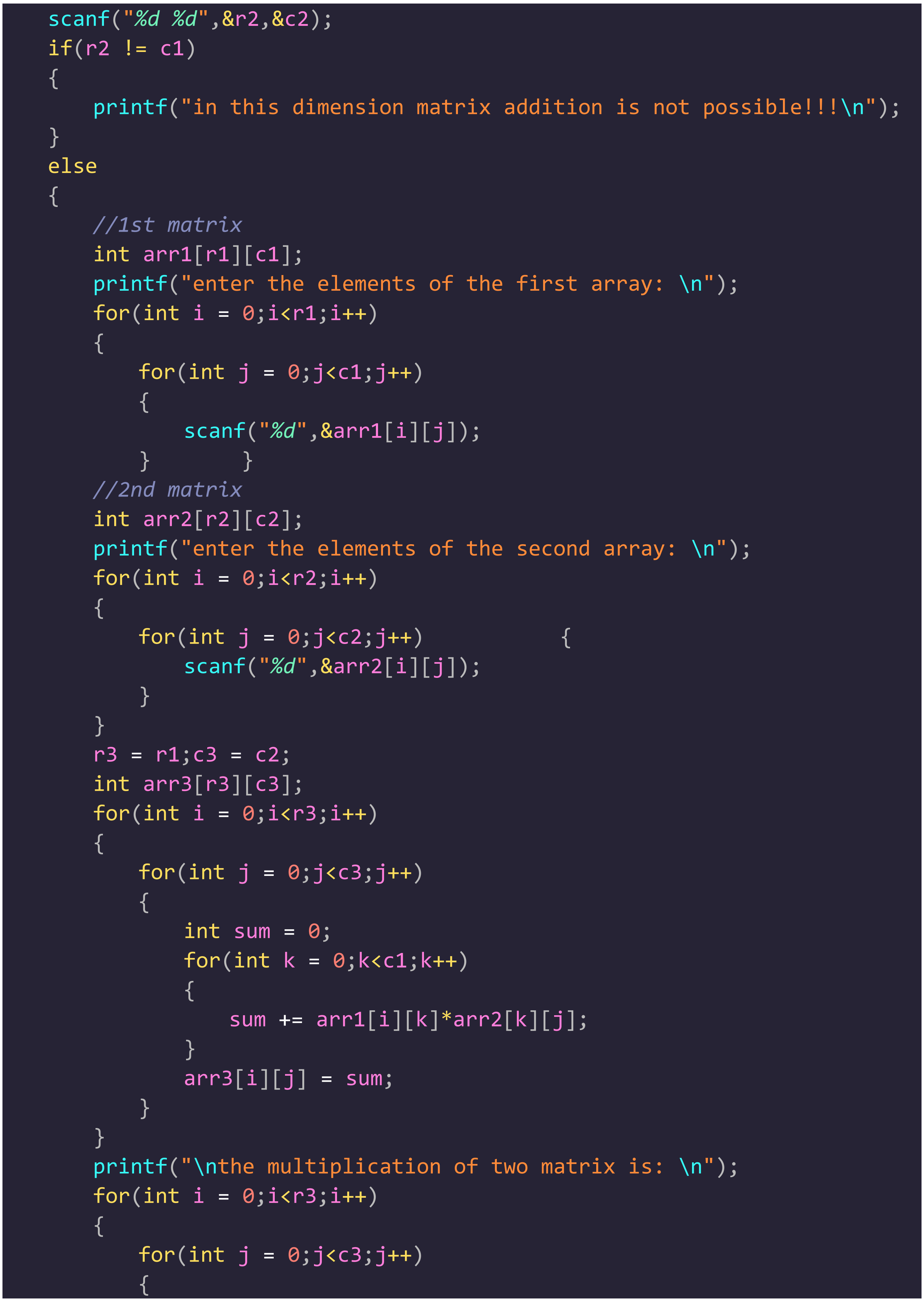
Problem Statement: Write a menu-driven program to implement matrix-addition, subtraction and multiplication.

Source code:

```c
#include<stdio.h>
#include<stdlib.h>

void addiiton()
{
    int r1,c1;
    int r2,c2;
    int r3,c3;
    // 1st matrix
    printf("enter the dimension of the first matrix(a b): \n");
    scanf("%d %d",&r1,&c1);
    printf("enter the dimension of the second matrix(a b): \n");
    scanf("%d %d",&r2,&c2);
    // 1st matrix
    if(r1 != r2 || c1 != c2)
    {
        printf("in this dimension matrix addition is not possible!!!\n");
    }
    else
    {
        // 1st matrix
        int arr1[r1][c1];
        printf("enter the elements of the first array: \n");
        for(int i = 0;i<r1;i++)
        {
            for(int j = 0;j<c1;j++)
            {
                scanf("%d",&arr1[i][j]);
            }
        }
        // 2nd matrix
        int arr2[r2][c2];
        printf("enter the elements of the second array: \n");
        for(int i = 0;i<r2;i++)
        {
            for(int j = 0;j<c2;j++)
            {
                scanf("%d",&arr2[i][j]);
            }
        }
        r3 = r1;c3 = c2;
        int arr3[r3][c3];
        for(int i = 0;i<r3;i++)
        {
            for(int j = 0;j<c3;j++)
            {
                arr3[i][j] = arr1[i][j]+arr2[i][j];
            }
        }
        printf("\nthe addition of two matrix is: \n");
        for(int i = 0;i<r3;i++)
        {
            for(int j = 0;j<c3;j++)
            {
                printf("%d ",arr3[i][j]);
            }
            printf("\n");
        }
    }
}
void subtraction()
{
    int r1,c1;
    int r2,c2;
    int r3,c3;
    // 1st matrix
    printf("enter the dimension of the first matrix(a b): \n");
    scanf("%d %d",&r1,&c1);
    printf("enter the dimension of the second matrix(a b): \n");
    scanf("%d %d",&r2,&c2);
    // 1st matrix
    if(r1 != r2 || c1 != c2)
    {
        printf("in this dimension matrix subtraction is not possible!!!\n");
    }
    else
    {
        // 1st matrix
        int arr1[r1][c1];
        printf("enter the elements of the first array: \n");
        for(int i = 0;i<r1;i++)
        {
            for(int j = 0;j<c1;j++)
```

```c
            {
                scanf("%d",&arr1[i][j]);
            }
        }
        // 2nd matrix
        int arr2[r2][c2];
        printf("enter the elements of the second array: \n");
        for(int i = 0;i<r2;i++)
        {
            for(int j = 0;j<c2;j++)
            {
                scanf("%d",&arr2[i][j]);
            }
        }
        r3 = r1;c3 = c2;
        int arr3[r3][c3];
        for(int i = 0;i<r3;i++)
        {
            for(int j = 0;j<c3;j++)
            {
                arr3[i][j] = arr1[i][j]-arr2[i][j];
            }
        }
        printf("\nthe subtraction of two matrix is: \n");
        for(int i = 0;i<r3;i++)
        {
            for(int j = 0;j<c3;j++)
            {
                printf("%d ",arr3[i][j]);
            }
            printf("\n");
        }
    }
}
void multiplication()
{
    int r3,c3;
    int r1,c1;
    int r2,c2;
    // 1st matrix
    printf("enter the dimension of the first matrix(a b): \n");
    scanf("%d %d",&r1,&c1);
    // 2nd matrix
    printf("enter the dimension of the second matrix(a b): \n");
```



```c
    scanf("%d %d",&r2,&c2);
    if(r2 != c1)
    {
        printf("in this dimension matrix addition is not possible!!!\n");
    }
    else
    {
        //1st matrix
        int arr1[r1][c1];
        printf("enter the elements of the first array: \n");
        for(int i = 0;i<r1;i++)
        {
            for(int j = 0;j<c1;j++)
            {
                scanf("%d",&arr1[i][j]);
            }        }
        //2nd matrix
        int arr2[r2][c2];
        printf("enter the elements of the second array: \n");
        for(int i = 0;i<r2;i++)
        {
            for(int j = 0;j<c2;j++)            {
                scanf("%d",&arr2[i][j]);
            }
        }
        r3 = r1;c3 = c2;
        int arr3[r3][c3];
        for(int i = 0;i<r3;i++)
        {
            for(int j = 0;j<c3;j++)
            {
                int sum = 0;
                for(int k = 0;k<c1;k++)
                {
                    sum += arr1[i][k]*arr2[k][j];
                }
                arr3[i][j] = sum;
            }
        }
        printf("\nthe multiplication of two matrix is: \n");
        for(int i = 0;i<r3;i++)
        {
            for(int j = 0;j<c3;j++)
            {
                printf("%d ",arr3[i][j]);
            }
            printf("\n");
        }
    }
}
int main()
{
    int choice;
    printf("\nMatrix Operations Menu:\n");
        printf("1. Matrix Addition\n");
        printf("2. Matrix Subtraction\n");
        printf("3. Matrix Multiplication\n");
        printf("4. Exit\n\n");
        printf("Enter your choice (1-4): ");
        scanf("%d", &choice);
    if(choice!=4)
    {
        if(choice == 1)
        {
            addiiton();
        }
        else if (choice == 2)
        {
            subtraction();
        }
        else if(choice == 3)
        {
            multiplication();
        }
    }
    else
        exit(1);
}
```

Output:

Case 1:

```
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Matrix Multiplication
4. Exit

Enter your choice (1-4): 1
```

```
enter the dimension of the first matrix(a b):
3 3
enter the dimension of the second matrix(a b):
3 3
enter the elements of the first array:
1 2 3
4 5 6
7 8 9
enter the elements of the second array:
1 2 3
1 2 3
1 2 3

the addition of two matrix is:
2 4 6
5 7 9
8 10 12
```

Case 2:

```
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Matrix Multiplication
4. Exit

Enter your choice (1-4): 2
enter the dimension of the first matrix(a b):
4 4
enter the dimension of the second matrix(a b):
4 4
enter the elements of the first array:
5 5 5 5
4 4 4 4
3 3 3 3
2 2 2 2
enter the elements of the second array:
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4

the subtraction of two matrix is:
4 4 4 4
```

```
2 2 2 2
0 0 0 0
-2 -2 -2 -2
```

Case 3:

```
Matrix Operations Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Matrix Multiplication
4. Exit

Enter your choice (1-4): 3
enter the dimension of the first matrix(a b):
3 4
enter the dimension of the second matrix(a b):
4 3
enter the elements of the first array:
1 2 3
1 2 3
1 2 3
1 2 3
enter the elements of the second array:
3 4 5 6
3 4 5 6
3 4 5 6

the multiplication of two matrix is:
34 33 28
37 33 37
37 42 43
```

Conclusion:

The provided C program offers a menu-driven interface for performing basic matrix operations, including addition, subtraction, and multiplication. It dynamically accepts user input for the dimensions and elements of two matrices, performs the chosen operation, and displays the result.

However, there are a few areas for improvement. The program could benefit from enhanced user guidance, such as displaying the matrix dimensions during input. The multiplication function appropriately checks the compatibility of matrix dimensions.

Assignment No. 21

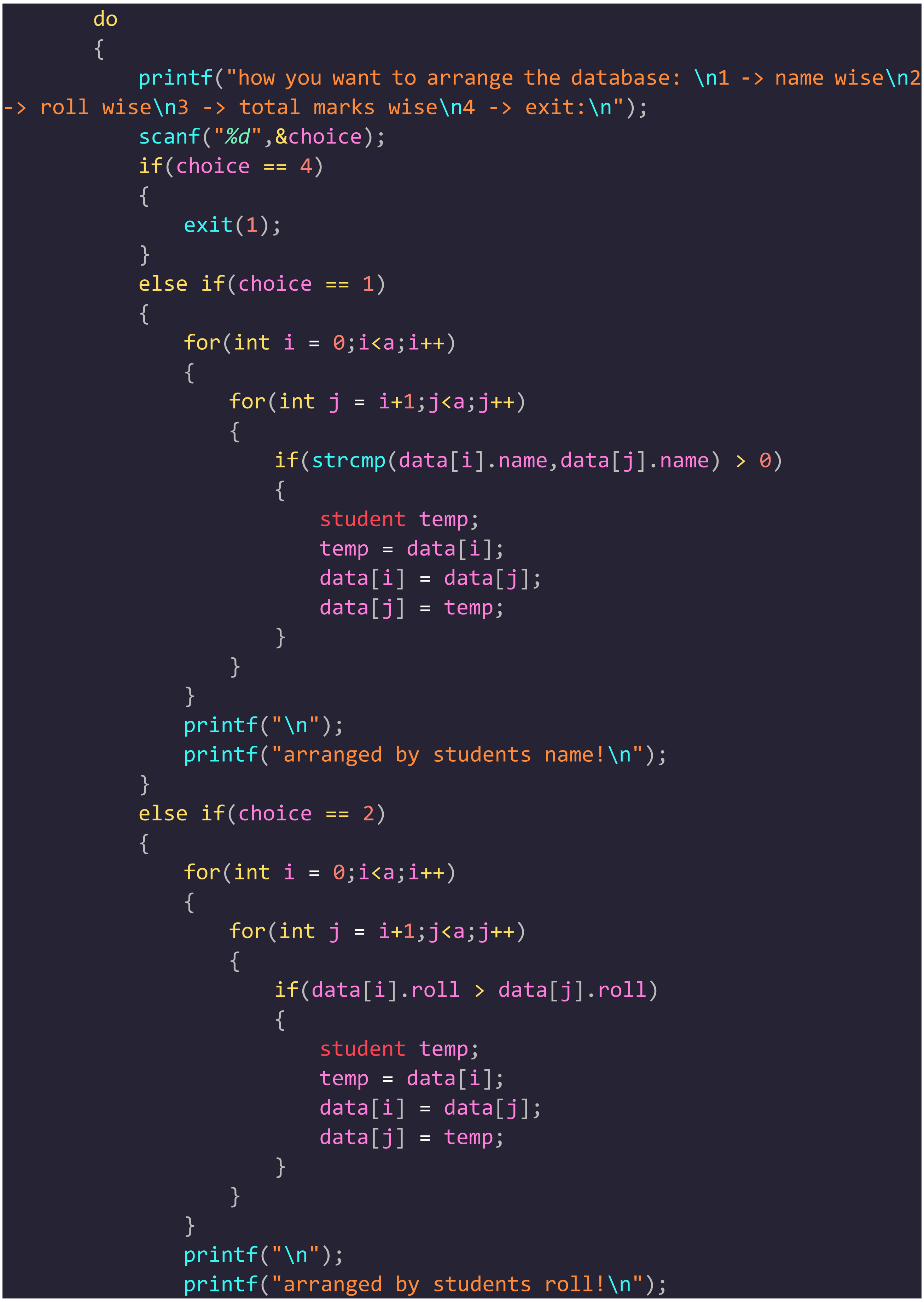
Problem Statement: Write a program to create a student database of N number of records. Each record consists of followings:

Student_Name, Student_Roll, Total_Marks, Subject
(Ex: Amit Dey, 13, 345, Computer Science) Arrange the records with respect to Student_Name, Student_Roll and Total_Marks respectively. Create the database using "structure" as well as "union" respectively.

Source code:

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

typedef struct student{
    char name[20];
    int roll;
    int total_marks;
    char subject[20];
}student;
int main()
{
    int a;
    printf("how many numbers of records do you want to save: \n");
    scanf("%d",&a);
    student data[a];
    if(a>0)
    {
        for(int i = 0;i<a;i++)
        {
            printf("record no %d\n",i+1);
            printf("student\'s name: ");
            scanf(" %[^\n]",data[i].name);
            printf("student\'s roll: ");
            scanf("%d",&data[i].roll);
            printf("total marks of the student: ");
            scanf("%d",&data[i].total_marks);
            printf("subject of the student: ");
            scanf(" %[^\n]",data[i].subject);
            printf("\n");
        }
        printf("\n");
        int choice;
```



```c
    do
    {
      printf("how you want to arrange the database: \n1 -> name wise\n2
-> roll wise\n3 -> total marks wise\n4 -> exit:\n");
      scanf("%d",&choice);
      if(choice == 4)
      {
        exit(1);
      }
      else if(choice == 1)
      {
        for(int i = 0;i<a;i++)
        {
          for(int j = i+1;j<a;j++)
          {
            if(strcmp(data[i].name,data[j].name) > 0)
            {
              student temp;
              temp = data[i];
              data[i] = data[j];
              data[j] = temp;
            }
          }
        }
        printf("\n");
        printf("arranged by students name!\n");
      }
      else if(choice == 2)
      {
        for(int i = 0;i<a;i++)
        {
          for(int j = i+1;j<a;j++)
          {
            if(data[i].roll > data[j].roll)
            {
              student temp;
              temp = data[i];
              data[i] = data[j];
              data[j] = temp;
            }
          }
        }
        printf("\n");
        printf("arranged by students roll!\n");
```

```
            }
            else if(choice == 3)
            {
                for(int i = 0;i<a;i++)
                {
                    for(int j = i+1;j<a;j++)
                    {
                        if(data[i].total_marks < data[j].total_marks)
                        {
                            student temp;
                            temp = data[i];
                            data[i] = data[j];
                            data[j] = temp;
                        }
                    }
                }
                printf("\n");
                printf("arranged by students total marks!\n");
            }

            for(int i = 0;i<a;i++)
            {
                printf("student\'s name: %s\n",data[i].name);
                printf("student\'s roll: %d\n",data[i].roll);
                printf("total marks of the student: %d\n",data[i].total_marks);
                printf("subject of the student: %s\n",data[i].subject);
                printf("\n");
            }
        }while(choice != 4);
    }
}
```

Output:

```
how many numbers of records do you want to save:

record no 1
student's name: amitav pal
student's roll: 12
total marks of the student: 56
subject of the student: mathematics

record no 2
```

```
student's name: arjun sen
student's roll: 67
total marks of the student: 34
subject of the student: biology

record no 3
student's name: biplab dey
student's roll: 4
total marks of the student: 12
subject of the student: bengali


how you want to arrange the database:
1 -> name wise
2 -> roll wise
3 -> total marks wise
4 -> exit:
1

arranged by students name!
student's name: amitav pal
student's roll: 12
total marks of the student: 56
subject of the student: mathematics

student's name: arjun sen
student's roll: 67
total marks of the student: 34
subject of the student: biology

student's name: biplab dey
student's roll: 4
total marks of the student: 12
subject of the student: bengali

How you want to arrange the database:
1 -> name wise
2 -> roll wise
3 -> total marks wise
4 -> exit:
2

arranged by students roll!
student's name: biplab dey
```

```
student's roll: 4
total marks of the student: 12
subject of the student: bengali

student's name: amitav pal
student's roll: 12
total marks of the student: 56
subject of the student: mathematics

student's name: arjun sen
student's roll: 67
total marks of the student: 34
subject of the student: biology

how you want to arrange the database:
1 -> name wise
2 -> roll wise
3 -> total marks wise
4 -> exit:
3

arranged by students total marks!
student's name: amitav pal
student's roll: 12
total marks of the student: 56
subject of the student: mathematics

student's name: arjun sen
student's roll: 67
total marks of the student: 34
subject of the student: biology

student's name: biplab dey
student's roll: 4
total marks of the student: 12
subject of the student: bengali

how you want to arrange the database:
1 -> name wise
2 -> roll wise
3 -> total marks wise
4 -> exit:
4
```

Conclusion:

The provided C program effectively manages a dynamic database of student records through a struct, allowing users to input, organize, and display information. Users can specify the number of records, input data for each student, and choose from various sorting options like name, roll number, or total marks. The program correctly employs sorting algorithms to arrange the records based on user preference. While the program successfully fulfills its intended purpose, there are areas for potential improvement.

Assignment No. 22

Problem Statement: Write a program to check a string (case sensitive) is palindrome or not.

Source code:

```c
#include<stdio.h>
#include<string.h>

void reverse(char arr1[],char arr2[])
{
    int length = strlen(arr1);
    int i;
    for(i = 0;i != length;i++)
    {
        arr2[i] = arr1[length-i-1];
    }
    arr2[i] = arr1[length];
}
int is_palindrome(char string[],char rev_string[])
{
    int i;
    for(i=0;string[i]!='\0';i++)
    {
        if (string[i] != rev_string[i])
        {
            return 0;
            break;
        }
    }
    return 1;
}
int main()
{
    char string[100];
    char rev_string[100];
    int check;
    printf("enter the string: ");
    gets(string);
    int length = strlen(string);
    reverse(string,rev_string);
    check = is_palindrome(string,rev_string);

    if (check == 1)
    {
        printf("%s is palindrome",string);
    }
    else if(check == 0)
    {
        printf("%s is not palindrome",string);
    }
}
```

Output:

Case 1:

```
enter the string: racecar
racecar is palindrome
```

Case 2:

```
enter the string: hello
hello is not palindrome
```

Conclusion:

The provided C program checks if a given string is a palindrome. It employs two functions: one to reverse the input string and another to compare the original string with its reversed counterpart. The program then outputs whether the input string is a palindrome or not. However, it relies on the deprecated gets function for string input, which may pose a security risk due to potential buffer overflow. Utilizing fgets for string input would be a safer alternative. Additionally, the program assumes the input string is null-terminated, and it may not handle cases where the input is not a valid string.

Assignment No. 23

Problem Statement: Write a program to sort a list of strings in ascending and descending order.

```c
#include<stdio.h>
#include<string.h>

int length(char *arr[],int len)
{
    return len;
}
void ascending(char **arr,int len)
{
    for(int i = 0;i<len;i++)
    {
        for(int j = (i+1);j<len;j++)
        {
            if (strcmp(arr[i],arr[j])>0)
            {
                char *k;
                k = arr[i];
                arr[i] = arr[j];
                arr[j] = k;
            }
        }
    }
}
void descending(char **arr,int len)
{
    for(int i = 0;i<len;i++)
    {
        for(int j = (i+1);j<len;j++)
        {
            if (strcmp(arr[i],arr[j])<0)
            {
                char *k;
                k = arr[i];
                arr[i] = arr[j];
                arr[j] = k;
            }
        }
    }
}
int main()
{
    char *arr[] = {"Aarav Sharma","Naina Patel","Arjun Singh","Aisha
Kapoor","Raj Verma","Ananya Das","Vikram Mehta","Diya Choudhury","Ravi
Kapoor","Priya Mishra"};
    int len = length(arr,sizeof(arr)/sizeof(arr[1]));
    int choice;
    printf("how do you want the list
represents\n1-->ascending\n2-->descending\n");
    scanf("%d",&choice);
    if(choice == 1)
        ascending(arr,len);
    else if(choice == 2)
        descending(arr,len);
    else
        printf("wrong input");
    for(int i = 0;i<len;i++)
    {
        printf("%s \n",arr[i]);
    }
}
```

Output:

Case 1:

```
how do you want the list represents
1-->ascending
2-->descending
1
Aarav Sharma
Aisha Kapoor
Ananya Das
Arjun Singh
Diya Choudhury
Naina Patel
Priya Mishra
Raj Verma
Ravi Kapoor
Vikram Mehta
```

Case 2:

```
how do you want the list represents
1-->ascending
2-->descending
2
Vikram Mehta
Ravi Kapoor
Raj Verma
Priya Mishra
Naina Patel
Diya Choudhury
Arjun Singh
Ananya Das
Aisha Kapoor
Aarav Sharma
```

Conclusion:

The provided C program takes an array of strings representing names, allows the user to choose between ascending and descending order, and then sorts and prints the names accordingly. The

sorting is done using the strcmp function for string comparison, and a simple swap mechanism is employed for rearranging the strings based on the user's choice of sorting order. The program concludes by printing the sorted list of names. Additionally, the program could be improved, such as input validation for the user's choice and potentially more robust sorting algorithms for larger datasets.

Assignment No. 24

Problem Statement: Write a program to read content of a “.txt” file and copy the content of the said file into another “.txt” file.

Source code:

```c
#include<stdio.h>
int main()
{
    FILE * fp;
    FILE * fs;
    char ch;
    fp = fopen("new_1.txt","r");
    fs = fopen("new_2.txt","w");
    if (fp == NULL || fs == NULL)
        printf("file is not opend");
    while(1)
    {
        ch = fgetc(fp);
        if (ch == EOF)
            break;
        else
            fputc(ch,fs);
    }
    printf("content is copied to another file!!");
    fclose(fp);fclose(fs);
}
```

Output:

```
content is copied to another file!!
```

Conclusion:

The provided C program reads the contents of a file named "new_1.txt" and writes the same

contents to another file named "new_2.txt". The program uses the fopen , fgetc , fputc , and fclose functions to achieve file handling operations. This is not a unique solution to solve this program; it could have been solved by other methods. It can be improved by some changes.