: C Programming Laboratory - 4.5CA151C01

Submitted By	
Student Name	Hansika
Roll No.	
Programme	
Semester	
Section	
Session/Batch	
Submitted To	
Faculty Name	

Manav Rachna International
Institute of Research and Studies
Deemed to be University under section
3 of the UGC Act, 1956

S.No.	Experiment Date	Aim of Experiment	Signature/Date	Grade
1	Friday, 15th Nov 2024	Arithmetic operations		
2	Monday, 25th Nov 2024	Calculate Gross Salary		
3	Friday, 15th Nov 2024	Sum and Reverse of a Three Digit Number		
4	Friday, 15th Nov 2024	Swap 2 Numbers		
5	Friday, 15th Nov 2024	Multiplication Table		
6	Friday, 15th Nov 2024	Greatest of Three Numbers		
7	Friday, 15th Nov 2024	Leap Year or Not		
8	Friday, 15th Nov 2024	Even or Odd		
9	Saturday, 16th Nov 2024	quadratic equation		
10	Tuesday, 19th Nov 2024	Weekday number using Switch-case		
11	Saturday, 16th Nov 2024	Using #define in C for String Manipulation		
12	Saturday, 23rd Nov 2024	First n Natural Numbers and their Sum		
13	Saturday, 16th Nov 2024	Small Factorials		

14	Saturday, 16th Nov 2024	Fibonacci series
15	Saturday, 16th Nov 2024	A Pyramid of Stars
16	Saturday, 16th Nov 2024	Palindrome or Not
17	Saturday, 16th Nov 2024	Maximum Number in an Array
18	Saturday, 16th Nov 2024	Search an Element in an Array
19	Tuesday, 26th Nov 2024	Addition of Two Matrices
20		Subtraction of Two Matrices
21		Multiplication of two Matrices
22	Tuesday, 26th Nov 2024	Matrix Transpose
23	Tuesday, 26th Nov 2024	Concatenate two strings without using built-in functions
24	Tuesday, 26th Nov 2024	Prime and Armstrong
25		Binary to Decimal and Decimal to Binary
26	Tuesday, 26th Nov 2024	Usage of Pointer

27	Tuesday, 26th Nov 2024	Reverse an Array In Place
28	Tuesday, 26th Nov 2024	Student Details using Structure
29	Wednesday, 27th Nov 2024	Add Two Distances (in inch-feet) Using Structures
30	Wednesday, 27th Nov 2024	Add two Complex Numbers

Experiment - 1

Experiment - 1: ***Arithmetic operations***

Name: Hansika
UUID:
Date of performance: 15th Nov 2024

Aim:
Write a C program to perform arithmetic operations like +,-,*,/,% on two input variables.

Input Format:

- The first line of input is an integer representing the value for first number
- The second line of input is an integer representing the value of second number

Output Format:

- The program prints the results of addition, subtraction, multiplication, division, and modulus each on a new line.

Note : For Division and Modulo operation, the value of num2 must be greater than 0

Add new line char \n at the end of output.

Tasks to be done:

Observations:

Program:

`arithmeticOperations.c`

```c
#include <stdio.h>
int main()  {
   int num1, num2;
   scanf("%d", &num1);
   scanf("%d", &num2);

   printf("%d\n", num1 + num2);
   printf("%d\n", num1 - num2);
   printf("%d\n", num1 * num2);

   if (num2 > 0) {
        printf("%d\n", num1 / num2);
        printf("%d\n", num1 % num2);

   }

   return 0;

}
```

Output:

Test case - 1
User Output
9
8
17
1
72
1
1

Test case - 2
User Output
1000
2
1002
998
2000
500
0

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 2

Experiment - 2: *Calculate Gross Salary*

Name: Hansika
UUID:
Date of performance: 25th Nov 2024

Aim:
Write the C program toInput the basic salary of an employee and calculate his/her gross salary based on the following conditions:

- If Basic Salary <= 10000, then HRA = 20% and DA = 80%
- If Basic Salary <= 20000, then HRA = 25% and DA = 90%
- IfBasic Salary > 20000, then HRA = 30% and DA = 95%

Gross Salary is the total of Basic Salary, HRA and DA.

Note: Output the grass salary up to two decimal points only.

Tasks to be done:

Observations:

Program:

salary.c

```c
#include <stdio.h>

int main() {
    float basic_salary, hra, da, Gross_salary;

    printf("Enter basic salary: ");
    scanf("%f" , &basic_salary);
    if (basic_salary< 10000) {
        hra =0.20 * basic_salary;
        da =0.90 * basic_salary;
    } else {
        hra = 0.30 * basic_salary;
        da = 0.95 * basic_salary;
    }
    Gross_salary = basic_salary + hra + da;
    printf("Gross salary: %.2f\n", Gross_salary);

    return 0;
}
```

Output:

Test case - 1
User Output
Enter basic salary:
35000
Gross salary: 78750.00

Test case - 2
User Output
Enter basic salary:
90000
Gross salary: 202500.00

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 3

Experiment - 3: ***Sum and Reverse of a Three Digit Number***

Name: Hansika
UUID:
Date of performance: 15th Nov 2024

Aim:
Write a C program to find sum and reverse of three digit number.

Input Format:
A single line of input contains a three-digit integer

Output Format:
The first line contains an integer representing the sum of the digits of the given integer
The second line contains an integer representing the reverse of the given integer
If the input is not a three-digit non negative number, print "Invalid input" on a single line.

Constraints:
100 <= num <= 999

Tasks to be done:

Observations:

Program:

`sumReverse.c`

```
#include <stdio.h>
int main() {
        int num, sum = 0, reverse = 0, digit;
        scanf("%d", &num);
        if(num < 100 || num >999)
{
    printf("Invalid input\n");
return 0;
}

int temp = num;
while (temp > 0) {
        digit = temp % 10;
        sum += digit;
        reverse = reverse * 10 + digit;
        temp /= 10;
}

printf("%d\n", sum);
printf("%d\n", reverse);

return 0;

}
```

Output:

Test case - 1
User Output
59
Invalid input

Test case - 2
User Output
100
1
1

Test case - 3
User Output
251
8
152

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 4

Experiment - 4: *Swap 2 Numbers*

Name: Hansika
UUID:
Date of performance: 15th Nov 2024

Aim:
Write a C program to swap two numbers without using a third variable.

Input Format:
The first line contains an integer representing the first number
The second line contains an integer representing the second number

Note: You just need to implement the logic to swap the two numbers without using third variable, printing the result has already been given.

Tasks to be done:

Observations:

Program:

`swapwithoutvar.c`

```
#include <stdio.h>

int main() {
    int num1, num2;
scanf("%d" , &num1);
scanf("%d" , &num2);


num1 = num1 + num2;
num2 = num1 - num2;
num1 = num1 - num2;


    printf("After swapping, first number is: %d\n", num1);
    printf("After swapping, second number is: %d", num2);

return 0;
}
```

Output:

Test case - 1
User Output
12
18
After swapping, first number is: 18
After swapping, second number is: 12

Test case - 2
User Output
90
56
After swapping, first number is: 56
After swapping, second number is: 90

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 5

Experiment - 5: *Multiplication Table*

Name: Hansika
UUID:
Date of performance: 15th Nov 2024

Aim:
Write a C program to print the multiplication table of a given number.

Input Format:
A single line of input contains an integer representing the number to which the multiplication table is to be printed

Output Format:
The multiplication table of the given number up to 10 values

Tasks to be done:

Observations:

Program:

multiplicationTable.c

```
#include<stdio.h>
int main() {
        int num;

        scanf("%d" , &num);
        for (int i =1; i <=10; i++) {
                printf("%d x %d = %d\n" ,num, i , num * i);

        }
        return 0;
}
```

Output:

Test case - 1
User Output
2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

Test case - 2
User Output
5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 6

Experiment - 6: *Greatest of Three Numbers*

Name: Hansika
UUID:
Date of performance: 15th Nov 2024

Aim:
Write a C program to display the greatest of three numbers.

Input Format
The program prompts the user to enter three integers in three lines.

Output Format
The program prints the greatest of the three integers.

Tasks to be done:

Observations:

Program:

greatestOfThree.c

```c
#include <stdio.h>
int main() {
        int num1,num2,num3;
    scanf("%d" , &num1);
    scanf("%d" , &num2);
    scanf("%d" , &num3);

    int greatest = num1;
    if (num2 > greatest) {
        greatest = num2;
    }
    if (num2 > greatest) {
        greatest = num2;
    }
    if (num3 > greatest) {
        greatest = num3;
    }
    printf("%d\n" , greatest);
        return 0;
}
```

Output:

Test case - 1
User Output
12
36
45
45

Test case - 2
User Output
91
25
46
91

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 7

Experiment - 7: *Leap Year or Not*

Name: Hansika
UUID:
Date of performance: 15th Nov 2024

Aim:
Write a C program to find whether the entered year is a leap year or not.

Input Format:
- Input consists of a four digit integer representing the year

Output Format:
- Print "**Leap year**" if the given is a leap year, otherwise print "**Not a leap year**".

Tasks to be done:

Observations:

Program:

`leapYear.c`

```c
#include<stdio.h>

int main() {
        int year;
        scanf("%d" , &year);

        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0 ))
        {
                printf("Leap year\n");
        } else {
                printf("Not a leap year\n");
        }
        return 0;
}
```

Output:

Test case - 1
User Output
2016
Leap year

Test case - 2
User Output
2011
Not a leap year

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 8

Experiment - 8: *Even or Odd*

Name: Hansika
UUID:
Date of performance: 15th Nov 2024

Aim:
Write a C program to check whether given number is even or odd.

Input Format:
Input is a single line of input contains a positive integer

Output Format:
Print "Even" if the number is even, otherwise print "Odd".

Tasks to be done:

Observations:

Program:

evenOdd.c

```
#include<stdio.h>

int main() {
        int num;
    scanf("%d" , &num);
    if (num % 2 == 0) {
        printf("Even\n");
    } else {
        printf("Odd\n");
    }
    return 0;
}
```

Output:

Test case - 1
User Output
45
Odd

Test case - 2
User Output
54
Even

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 9

Experiment - 9: *quadratic equation*

Name: Hansika
UUID:
Date of performance: 16th Nov 2024

Aim:
Write a C program that takes three coefficients *a*,*b*, and c of a quadratic equation $ax^2+bx+c = 0$ as input and computes all possible roots.

Note: Print the output up to 2 decimals

Example Input:
Enter coefficient a: 4
Enter coefficient b: 7
Enter coefficient c: 2

Example output:
Root 1: -0.36
Root 2: -1.39

Tasks to be done:

Observations:

Program:

quadratic.c

```
#include<stdio.h>
#include<math.h>

int main () {
        float a, b ,c;
        printf("Enter coefficient a: ");
        scanf("%f" ,&a);
        printf("Enter coefficient b: ");
        scanf("%f" ,&b);
        printf("Enter coefficient c: ");
        scanf("%f" ,&c);

   float discriminant = b * b - 4 * a* c;

    if(discriminant < 0 && discriminant > -0.0001) {
        discriminant = 0;
    }
    if (discriminant >= 0){

        float root1 = (-b + sqrt (discriminant)) / (2 * a);
        float root2 = (-b - sqrt (discriminant)) / (2 * a);

        printf("Root 1: %.2f\n" , root1);
        printf("Root 2: %.2f\n" , root2);
    } else {
        printf("Roots are imaginary. \n");
    }

    return 0;
}
```

Output:

Test case - 1
User Output
Enter coefficient a:
4
Enter coefficient b:
7
Enter coefficient c:
2
Root 1: -0.36
Root 2: -1.39

Test case - 2
User Output
Enter coefficient a:
-3
Enter coefficient b:
6
Enter coefficient c:
2

```
Root 1: -0.29
Root 2: 2.29
```

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 10

Experiment - 10: ***Weekday number using Switch-case***

Name: Hansika
UUID:
Date of performance: 19th Nov 2024

Aim:
Write a program to read the **weekday number** from the standard input and print the **weekday name** using **switch-case**.

Assume that the weekdays are provided with the below numbers:

- Sunday ⟹ 0
- Monday ⟹ 1
- Tuesday ⟹ 2
- Wednesday ⟹ 3
- Thursday ⟹ 4
- Friday ⟹ 5
- Saturday ⟹ 6

Input Format:

- A single line of input contains a positive integer representing the weekday number

Output Format:

- The corresponding day of the week based on the above given criteria.
- If the given number is not a valid weekday number, print **"Invalid weekday number"**

Tasks to be done:

Observations:

Program:

`Weekday.c`

```
#include <stdio.h>
int main() {
        int weekdayNumber;
        scanf("%d" , &weekdayNumber);
        switch (weekdayNumber) {
                case 0:
                printf("Sunday\n");
                break;
                case 1:
                printf("Monday\n");
                break;
                case 2:
                printf("Tuesday\n");
                break;
                case 3:
                printf("Wednesday\n");
            break;
                case 4:
                printf("Thursday\n");
                break;
                case 5:
                printf("Friday\n");
                break;
                case 6:
                printf("Saturday\n");
                break;
                default:
                printf("Invalid weekday number\n");
                break;
        }
        return 0;
}
```

Output:

Test case - 1
User Output
6
Saturday

Test case - 2
User Output
9
Invalid weekday number

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 11

Experiment - 11: ***Using #define in C for String Manipulation***

Name: Hansika
UUID:
Date of performance: 16th Nov 2024

Aim:
Write a C program that uses a macro to define a constant string and demonstrates string manipulation operations. Implement the program with the following features:

- Define a macro SAMPLE_STRING with a user-defined string literal "Hello, World!".
- Print the defined sample string.
- Calculate and print the length of the sample string.
- Print the sample string in reverse order.

Tasks to be done:

Observations:

Program:

StringManipulation.c

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

#define SAMPLE_STRING "Hello, World!"
int main() {
        printf("%s\n",SAMPLE_STRING);
        int length = strlen(SAMPLE_STRING);
        for (int i = length - 1; i >= 0; i--) {
                putchar(SAMPLE_STRING[i]);
}
printf("\n");

        return 0;
}
```

Output:

Test case - 1
User Output
Hello, World!
!dlroW ,olleH

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 12

Experiment - 12: ***First n Natural Numbers and their Sum***

Name: Hansika
UUID:
Date of performance: 23rd Nov 2024

Aim:
Write a C program to display first n natural numbers and their sum.

Input Format:
A single line of input contains an integer representing the value of n

Output Format:
The first line contains n space separated elements representing the first n natural numbers
The second line contains the sum of the first n natural numbers

Note: For every output, the program must navigate to next line using '\n'

Tasks to be done:

Observations:

Program:

`firstNNaturals.c`

```
#include<stdio.h>
int main() {
        int n, sum = 0;
        scanf("%d" , &n);

        for(int i = 1; i <= n; i++) {
                printf("%d" , i);
                sum += i;

                if (i < n) {
                        printf(" ");
                }
   }
printf(" \n");

printf("%d\n" , sum);

return 0;
}
```

Output:

Test case - 1
User Output
5
1 2 3 4 5
15

Test case - 2
User Output
16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
136

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 13

Experiment - 13: *Small Factorials*

Name: Hansika
UUID:
Date of performance: 16th Nov 2024

Aim:
Write a C program to calculate the factorial of positive integers without using recursion.

Input Format:
Single line of input contains an integer N representing the value to find the factorial

Output Format:
Print the factorial result of N

Tasks to be done:

Observations:

Program:

`factorial.c`

```
#include<stdio.h>

int main() {
        int N;
        unsigned long long factorial = 1;
        scanf("%d" , &N);

    for (int i = 1; i <= N; i++) {
                        factorial *= i;
        }
    printf("%llu\n" , factorial);
        return 0;
        }
```

Output:

Test case - 1
User Output
2
2

Test case - 2
User Output
3
6

Test case - 3
User Output
4
24

Test case - 4
User Output
5
120

Test case - 5
User Output
1
1

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 14

Experiment - 14: *Fibonacci series*

Name: Hansika
UUID:
Date of performance: 16th Nov 2024

Aim:
Write a C program that prints the first N numbers of the Fibonacci series.

Input Format:
A single line contains an integer representing the value of N

Output Format:
First N numbers of the Fibonacci series are separated by space

Tasks to be done:

Observations:

Program:

`fibonacciSeries.c`

```
#include<stdio.h>
int main() {
        int N;

        scanf("%d", &N);
        unsigned long long a = 0 , b = 1;
        for(int i = 0; i < N; i++) {
                printf("%llu" , a);
                unsigned long long next = a + b;

                a = b;
                b = next;
                if (i < N -1) {
                        printf(" ");


                }
        }
        printf(" \n");

        return 0;
}
```

Output:

Test case - 1
User Output
5
0 1 1 2 3

Test case - 2
User Output
8
0 1 1 2 3 5 8 13

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 15

Experiment - 15: *A Pyramid of Stars*

Name: Hansika
UUID:
Date of performance: 16th Nov 2024

Aim:
Write a C program to print stars sequence in pyramid form.

Input Format:
A single line of input contains an integer representing the total number of rows of the pyramid

Output Format:
Output the pyramid of stars with n rows.

Tasks to be done:

Observations:

Program:

`pyramidOfStars.c`

```
#include<stdio.h>

int main()
{
        int n, i, j, space;

        scanf("%d",&n);
        for (i = 1; i <=n; i++){
                for (space = 1; space <= n -i;
                space++) {
                        printf(" ");
                }
                for(j = 1; j<=2*i - 1; j++){
                        printf("*");

                }
                printf("\n");
        }
        return 0;
}
```

Output:

Test case - 1
User Output
5
*
***
*****
****
****

Test case - 2
User Output
4
*
***
*****
****

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 16

Experiment - 16: *Palindrome or Not*

Name: Hansika
UUID:
Date of performance: 16th Nov 2024

Aim:
Write a C program to check whether the given number is palindrome or not.

Input Format:
A single line of input contains an integer.

Output Format:
If the given number is palindrome, then print "{number} is palindrome", otherwise, print "{number} is not palindrome".

Tasks to be done:

Observations:

Program:

palindrome.c

```c
// Type Content here...
#include<stdio.h>

int main() {
        int num, reversed = 0, remainder , original;

        scanf("%d" , &num);
        original = num;

        while (num != 0) {
                remainder = num % 10;
                reversed = reversed * 10 + remainder;
                num /= 10;
    }
        if (original == reversed) {
                printf("%d is palindrome\n" , original);
        } else {
                printf("%d is not palindrome\n" , original);
        }
    return 0;
}
```

Output:

Test case - 1
User Output
121
121 is palindrome

Test case - 2
User Output
134
134 is not palindrome

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 17

Experiment - 17: ***Maximum Number in an Array***

Name: Hansika
UUID:
Date of performance: 16th Nov 2024

Aim:
Write a C program to find the maximum number in an array.

Input Format:
The first line contains an integer n representing the total number of elements in the array
The next n lines contain an integer representing the elements of the array

Output Format:
The maximum element of the array in the format "Largest element: {number}"

Tasks to be done:

Observations:

Program:

largest.c

```c
#include <stdio.h>

int main() {
        int n, i, max;
        printf("n: ");
        scanf("%d" , &n);

        int arr[n];

        for (i = 0; i < n; i++) {
                scanf("%d" , &arr[i]);
        }
        max = arr[0];

        for(i =1; i < n; i++) {
                if (arr[i] > max) {
                        max = arr[i];
                }
        }
        printf("Largest element: %d\n" , max);
        return 0;
}
```

Output:

Test case - 1
User Output
n:
3
4
5
6
Largest element: 6

Test case - 2
User Output
n:
5
9
45
12
10
3
Largest element: 45

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 18

Experiment - 18: ***Search an Element in an Array***

Name: Hansika
UUID:
Date of performance: 16th Nov 2024

Aim:
Write a C program to search for an element in an array. Return the position of the user-given element if it is found in the array otherwise display that it is **not found.**

Input Format:
The first line contains an integer n representing the size of the array
The second line contains n space separated integers representing the elements of the array
The third line contains an integer s, representing the element to be searched

Output Format:
If the search element is found, print "{search_element} found at position {position}", otherwise print "{search_element} not found"

Tasks to be done:

Observations:

Program:

searchEle.c

```c
#include <stdio.h>
int main() {
        int n, i, search_element, position = -1;
        printf("Enter size: ");
        scanf("%d" , &n);

        int arr[n];
        printf("Enter elements: ");
        for (i = 0; i < n; i++)
        scanf("%d" , &arr[i]);

        printf("Enter search key: ");
        scanf("%d" , &search_element);

        for (i = 0; i < n; i++) {
                if (arr[i] == search_element) {
                        position = i + 1;
                break;
                }

        }
        if(position != -1) {
                printf("%d found at position %d\n" , search_element , position);
        } else {
                printf("%d not found\n", search_element);
        }
        return 0;
}
```

Output:

Test case - 1
User Output
Enter size:
6
Enter elements:
11 22 33 44 55 66
Enter search key:
33
33 found at position 3

Test case - 2
User Output
Enter size:
6
Enter elements:
11 22 33 44 55 66
Enter search key:
75
75 not found

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 19

Experiment - 19: *Addition of Two Matrices*

Name: Hansika
UUID:
Date of performance: 26th Nov 2024

Aim:
Write a C program to perform the addition of two matrices

Input Format:

- The first line contains two space separated integers, row & col, representing the number of rows and columns of each matrix
- The second line contains row * col number of space separated integers representing the elements of matrix 1
- The last line contains row * col number of space separated integers representing the elements of matrix 2

Output Format:

- Print row number of lines with col number of space separated elements representing the elements of sum matrix

Note: Addition of two matrices can only be done when the dimensions of both matrices are same, so we are taking the same dimensions for both matrices

Tasks to be done:

Observations:

Program:

addTwoMatrices.c

```c
#include<stdio.h>

int main() {
        int rows, cols;

        scanf("%d %d" , &rows, &cols);

        int matrix1[rows][cols], matrix2[rows][cols], sum[rows][cols];

        for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                        scanf("%d" ,&matrix1[i][j]);

                }
        }
        for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                        scanf("%d" , &matrix2[i][j]);
                }
        }
        for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                        sum[i][j] = matrix1[i][j] + matrix2[i][j];

                }
        }
        for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                        printf("%d " , sum[i][j]);

                }
                printf("\n");
        }
        return 0;
}
```

Output:

Test case - 1
User Output
2 2
1 2 3 4
5 6 7 8
6 8
10 12

Test case - 2
User Output
1 2
1 2
9 8
10 10

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 22

Experiment - 22: *Matrix Transpose*

Name: Hansika
UUID:
Date of performance: 26th Nov 2024

Aim:
Write a C program to generate the transpose of a matrix.

Input Format:

- The first line of input should contain two integers separated by a space. These integers represent the number of rows (n) and columns (m) of the matrix.
- The next n lines each contain m integers separated by spaces. These integers represent the elements of the matrix.

Output format:

- The program prints the original matrix in a grid format where each row is on a new line and each element is separated by a space.
- The program prints the transpose of the matrix. The transpose is formed by swapping rows with columns, so each column of the original matrix becomes a row in the transposed matrix.

Tasks to be done:

Observations:

Program:

`matrixTranspose.c`

```
#include<stdio.h>

int main () {
        int rows, cols;
        scanf("%d %d" , &rows , &cols);

        int matrix[rows][cols], transpose[cols][rows];

        for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j ++) {
                        scanf("%d" , &matrix[i][j]);

                }
        }

        printf("Original matrix:\n");
        for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++){
                        printf("%d" , matrix[i][j]);
                        if (j < cols -1) printf(" ");

                }
                printf(" \n");
        }

            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                        transpose[j][i] = matrix[i][j];

                }
            }
            printf("Transpose of the matrix:\n");
            for (int i = 0; i < cols; i++) {
                for (int j = 0; j < rows; j++) {
                        printf("%d" , transpose[i][j]);
                        if (j < rows -1) printf(" ");


                }
                printf(" \n");

            }
            return 0;
}
```

Output:

Test case - 1
User Output
3 3
1 2 3
4 5 6
7 8 9
Original matrix:
1 2 3

```
4 5 6
7 8 9
Transpose of the matrix:
1 4 7
2 5 8
3 6 9
```

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 23

Experiment - 23: ***Concatenate two strings without using built-in functions***

Name: Hansika
UUID:
Date of performance: 26th Nov 2024

Aim:
Write a C program to concatenate two strings without using **strcat** function.

Input Format:
- The first line contains the first string
- The second line contains the second string

Output Format:
- Output the concatenated string

Tasks to be done:

Observations:

Program:

`ConcatenateTwoStrings.c`

```
#include<stdio.h>
int main() {
        char str1[100], str2[100], result[200];
        int i = 0, j = 0;

        scanf("%s" , str1);
        scanf("%s" , str2);

        while (str1[i] != '\0') {
                result[i] = str1[i];
                i++;


        }

        while (str2[j] != '\0') {
                result[i] = str2[j];
                i++;
        j++;
}
result[i] = '\0';

printf("%s\n" , result);

return 0;
}
```

Output:

Test case - 1
User Output
Programming
Languages
ProgrammingLanguages

Test case - 2
User Output
Morning
SunShine
MorningSunShine

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 24

Experiment - 24: *Prime and Armstrong*

Name: Hansika
UUID:
Date of performance: 26th Nov 2024

Aim:
Write a C program that includes two functions: one to check if a given number is a prime number and another to check if the given number is an Armstrong number. The program should prompt the user to enter a number and then use these functions to determine and display whether the number is prime and whether it is an Armstrong number.

Input Format:
A single integer input n representing the number to be checked.

Output Format:
The first line should indicate whether the number is a prime number.
The second line should indicate whether the number is an Armstrong number.

Tasks to be done:

Observations:

Program:

PrimeArmstrong.c

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

// Function to check if a number is prime
int is_prime(int num) {
        if (num <= 1)
return 0;
for (int i = 2; i * i <= num; i++) {
        if (num % i == 0)
        return 0;
}
return 1;
}

// Function to check if a number is an Armstrong number
int is_armstrong(int num) {
        if (num >= 0&& num <= 9)
        return 1;
        int original = num, sum = 0;
        while (num > 0) {
                int digit = num % 10; sum += digit * digit * digit; num/= 10;
        }
        return sum == original;


}

int main() {
    int n;

    // Input from the user
    scanf("%d", &n);

    // Check if the number is prime
    if (is_prime(n))
        printf("%d is prime\n", n);
    else
        printf("%d is not prime\n", n);

    // Check if the number is an Armstrong number
    if (is_armstrong(n))
        printf("%d is Armstrong\n", n);
    else
        printf("%d is not Armstrong\n", n);

    return 0;
}
```

Output:

Test case - 1
User Output
153
153 is not prime
153 is Armstrong

Test case - 2
User Output
7
7 is prime
7 is Armstrong

Test case - 3
User Output
10
10 is not prime
10 is not Armstrong

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 26

Experiment - 26: *Usage of Pointer*

Name: Hansika
UUID:
Date of performance: 26th Nov 2024

Aim:
Write a C program to demonstrate the use of pointers. The program should take an integer input from the user and then print both the value and the address of the entered integer. This should be done using direct variable access as well as through a pointer

Input Format:
- The program should prompt the user to enter an integer.

Output Format:
- Print the value of the integer directly and print the value of the integer using the pointer in separate lines.

Note:
- The code snippet given is partially filled in. Your task is to complete the missing parts to ensure the program works as intended

Tasks to be done:

Observations:

Program:

valueOfPointer.c

```c
#include <stdio.h>
int main() {
    int num;
    int *ptr; // Declaration of pointer variable


    scanf("%d", &num      ); // Taking input from the user


    ptr = &num; // Assigning address of 'num' to 'ptr'


    printf("Value of num: %d\n",   num        );

    printf("Value of num using pointer: %d\n",  *ptr    ); // Dereferencing pointer to
access value

    return 0;
}
```

Output:

Test case - 1
User Output
10
Value of num: 10
Value of num using pointer: 10

Test case - 2
User Output
20
Value of num: 20
Value of num using pointer: 20

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 27

Experiment - 27: ***Reverse an Array In Place***

Name: Hansika
UUID:
Date of performance: 26th Nov 2024

Aim:
Write a C function that reverses the elements of an array in place. The function should accept a pointer to the array and the size of the array and should modify the array directly.

Input Format:
The first input line contains an integer n, representing the number of elements in the array.
The second line contains n space-separated integers representing the elements of the array.

Output Format:
Output the elements of the reversed array in a single line, space separated.

Tasks to be done:

Observations:

Program:

`reverseInPlace.c`

```c
#include <stdio.h>

void reverseArray(int *arr, int size) {
    int temp, start = 0, end = size -1;
    while(start < end) {
        temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;


        start++;
        end--;
    }



}

int main() {
    int n;

    // Input the size of the array
    scanf("%d", &n);

    int arr[n];

    // Input the elements of the array
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Reverse the array
    reverseArray(arr, n);

    // Output the reversed array
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

    return 0;
}
```

Output:

Test case - 1
User Output
5
1 2 3 4 5
5 4 3 2 1

Test case - 2
User Output
6

15 48 36 21 48 79
79 48 21 36 48 15

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 28

Experiment - 28: ***Student Details using Structure***

Name: Hansika
UUID:
Date of performance: 26th Nov 2024

Aim:
Write a C program to create a structure named Student that contains the following fields:

- Roll No.
- Name
- Class
- Year
- Total Marks

The program should allow the user to input details for a student and then display these details.

Input Format:
An integer representing the Roll No (int).
A string representing the Name (array).
An integer representing the Class (int).
An integer representing the Year (int).
A float representing the Total Marks (float).

Output Format:
Display the details of the student in a formatted manner.

Tasks to be done:

Observations:

Program:

stuDetails.c

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

// Define the Student structure
struct Student {


    int rollNo;;
    char name[20];
    int classs;
    int year;
    float totalMarks;




};

int main() {
    struct Student student;
  scanf("%d" , &student.rollNo);
    scanf("%s" , &student.name);
    scanf("%d" , &student.classs);
    scanf("%d" , &student.year);
    scanf("%f" , &student.totalMarks);
 // write your code here...




    printf("Student Details:\n");
    printf("Roll No: %d\n", student.rollNo);
    printf("Name: %s\n", student.name);
    printf("Class: %d\n", student.classs);
    printf("Year: %d\n", student.year);
    printf("Total Marks: %.2f\n", student.totalMarks);

    return 0;
}
```

Output:

Test case - 1
User Output
103
Aoemon
8
2022
35
Student Details:
Roll No: 103
Name: Aoemon
Class: 8
Year: 2022

Total Marks: 35.00

Test case - 2
User Output
27
Aeogon
9
2023
30
Student Details:
Roll No: 27
Name: Aeogon
Class: 9
Year: 2023
Total Marks: 30.00

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 29

Experiment - 29: ***Add Two Distances (in inch-feet) Using Structures***

Name: Hansika
UUID:
Date of performance: 27th Nov 2024

Aim:
Write a C Program to add two distances (in inch-feet) system using structures.

Input Format:
The first line contains two space separated elements representing the feet (integer) and inches (float) respectively of first distance.
The second line contains two space separated elements representing the feet (integer) and inches (float) respectively of second distance.

Output Format:
Output the sum of the two distances in the format "X ft Y inch" where the X is the value of feet and Y is the value of inches after addition.

Note: The value of Y in the output should be restricted to 1 decimal point only.

Tasks to be done:

Observations:

Program:

Distance.c

```c
#include <stdio.h>

struct Distance       {
        int feet;
        float inch;


};

int main() {
             struct Distance d1 , d2, sum;
             scanf("%d %f" , &d1.feet , &d1.inch);
             scanf("%d %f" , &d2.feet , &d2.inch);

             sum.inch = d1.inch + d2.inch;
             sum.feet = d1.feet + d2.feet;

             if(sum.inch >= 12.0) {
          sum.feet += (int) (sum.inch/12);
              sum.inch = (float) ((int)sum.inch%12) + (sum.inch - (int)sum.inch);

             }

             printf("%d f %.1f inch\n"  , sum.feet , sum.inch);




        return 0;
}
```

Output:

Test case - 1
User Output
34 2.4
23 8.6
57 f 11.0 inch

Test case - 2
User Output
54 3.65
21 2.3
75 f 5.9 inch

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes:

Experiment - 30

Experiment - 30: ***Add two Complex Numbers***

Name: Hansika
UUID:
Date of performance: 27th Nov 2024

Aim:
Write a C program to add two complex numbers by passing structure to a function.

Input Format:
The first line contains two space separated elements representing the real (float) and imaginary (float) values of first complex number respectively.
The second line contains two space separated elements representing the real (float) and imaginary (float) values of second complex number respectively.

Output Format:
Output the addition result in the format "**r + qi**" where r refers to the real value and q refers to the imaginary value.

Tasks to be done:

Observations:

Program:

`AddComplexNumbers.c`

```c
#include <stdio.h>

typedef struct      {
        float real;
        float imag;
        // initialize required variables

} complex;

complex add(complex n1, complex n2) {
        // write your code here
        complex result;
        result.real = n1.real + n2.real;
        result.imag = n1.imag + n2.imag;
        return result;

}

int main() {
    complex n1, n2, result;

    scanf("%f %f", &n1.real, &n1.imag);
    scanf("%f %f", &n2.real, &n2.imag);

    result = add(n1, n2);

    printf("%.1f + %.1fi", result.real, result.imag);
    return 0;
}
```

Output:

Test case - 1
User Output
1.2 3.1
1.3 5.2
2.5 + 8.3i

Test case - 2
User Output
3.2 4.1
6.2 4.8
9.4 + 8.9i

Result:

Thus the above program is executed successfully and the output has been verified

Learning Outcomes: