2. Decision and loops:

v) C program to display Armstrong number between two intervals.

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

int main() {
    int start, end, num, originalNum, remainder, n = 0, result = 0;

    printf("Enter the starting and ending interval: ");
    scanf("%d %d", &start, &end);
    printf("Armstrong Numbers between %d and %d are: ", start, end);
    for (num = start; num <= end; num++) {
        originalNum = num;

        // Count digits
        while (originalNum != 0) {
            originalNum /= 10;
            ++n;
        }

        originalNum = num;
        while (originalNum != 0) {
            remainder = originalNum % 10;
            result += pow(remainder, n);
            originalNum /= 10;
        }

        if (result == num) {
            printf("%d ", num);
        }

        n = 0;
        result = 0;
    }

    printf("\n");

    return 0;
}
```

Output:

```
Enter the starting and ending interval: 1
100
Armstrong Numbers between 1 and 100 are: 1 2 3 4 5 6 7 8 9
```

3. Function and recursion:

b) C program to check prime or Armstrong number using user defined function.

```c
// C program to check prime or Armstrong number using user defined function.

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

int isPrime(int num) {
    if (num <= 1) {
        return 0; // Not prime
    }

    for (int i = 2; i <= sqrt(num); i++) {
        if (num % i == 0) {
            return 0; // Not prime
        }
    }

    return 1; // Prime
}

```

```c
int isArmstrong(int num) {
    int originalNum = num, sum = 0, digit, numDigits = 0;

    // Count the number of digits
    while (originalNum > 0) {
        originalNum /= 10;
        numDigits++;
    }

    originalNum = num;

    while (originalNum > 0) {
        digit = originalNum % 10;
        sum += pow(digit, numDigits);
        originalNum /= 10;
    }

    return sum == num;
}
```

```c
int main() {
    int num;

    printf("Enter a positive integer: ");
    scanf("%d", &num);

    if (isPrime(num)) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    if (isArmstrong(num)) {
        printf("%d is an Armstrong number.\n", num);
    } else {
        printf("%d is not an Armstrong number.\n", num);
    }

    return 0;
}
```

Output:

```
Enter a positive integer: 153
153 is not a prime number.
153 is an Armstrong number.
```

e) C program to find GCD using recursion.

```c
#include <stdio.h>

int gcd(int a, int b) {
    if (b == 0) {
        return a;
    } else {
        return gcd(b, a % b);
    }
}

int main() {
    int num1, num2, result;

    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    result = gcd(num1, num2);

    printf("GCD of %d and %d is %d\n", num1, num2, result);

    return 0;
}
```

Output:

```
Enter two numbers: 24
6
GCD of 24 and 6 is 6
```

h) C program to reverse a sentence using recursion.

```c
#include <stdio.h>
void reverseSentence();
int main() {
    printf("Enter a sentence: ");
    reverseSentence();
    return 0;
}

void reverseSentence() {
    char c;
    scanf("%c", &c);
    if (c != '\n') {
        reverseSentence();
        printf("%c", c);
    }
}
```

Output:

```
Enter a sentence: hello
olleh
```

4. Array and pointers:

a) C program to calculate average using arrays.

```c
#include <stdio.h>
int main() {
    int n, i;
    float num[100], sum = 0.0, avg;

    printf("Enter the numbers of elements: ");
    scanf("%d", &n);

    while (n > 100 || n < 1) {
        printf("Error! number should in range of (1 to 100).\n");
        printf("Enter the number again: ");
        scanf("%d", &n);
    }

    for (i = 0; i < n; ++i) {
        printf("%d. Enter number: ", i + 1);
        scanf("%f", &num[i]);
        sum += num[i];
    }

    avg = sum / n;
    printf("Average = %.2f", avg);
    return 0;
}
```

Output

```
Enter the numbers of elements: 6
1. Enter number: 45.3
2. Enter number: 67.5
3. Enter number: -45.6
4. Enter number: 20.34
5. Enter number: 33
6. Enter number: 45.6
Average = 27.69
```

c) C program to count no. of positive numbers, negative numbers and zeros in the array.

```c
#include <stdio.h>

int main() {
    int n, i, positiveCount = 0, negativeCount = 0, zeroCount = 0;

    printf("Enter the number of elements: ");
    scanf("%d", &n);

    int arr[n];

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

        if (arr[i] > 0) {
            positiveCount++;
        } else if (arr[i] < 0) {
            negativeCount++;
        } else {
            zeroCount++;
        }
    }

    printf("\nPositive numbers: %d\n", positiveCount);
    printf("Negative numbers: %d\n", negativeCount);
    printf("Zeros: %d\n", zeroCount);

    return 0;
}

```

Output:

```
Enter the number of elements: 6
Enter the elements:
1
0
6
-8
0
-7

Positive numbers: 2
Negative numbers: 2
Zeros: 2
```

f) C program to multiply 2 Matrix using multi-dimensional arrays.

```c
#include <stdio.h>

// function to get matrix elements entered by the user
void getMatrixElements(int matrix[][10], int row, int column) {

   printf("\nEnter elements: \n");

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

// function to multiply two matrices
void multiplyMatrices(int first[][10],
                      int second[][10],
                      int result[][10],
                      int r1, int c1, int r2, int c2) {

   // Initializing elements of matrix mult to 0.
   for (int i = 0; i < r1; ++i) {
      for (int j = 0; j < c2; ++j) {
         result[i][j] = 0;
      }
   }

   // Multiplying first and second matrices and storing it in result
   for (int i = 0; i < r1; ++i) {
      for (int j = 0; j < c2; ++j) {
         for (int k = 0; k < c1; ++k) {
            result[i][j] += first[i][k] * second[k][j];
         }
      }
```

```
    }
}

// function to display the matrix
void display(int result[][10], int row, int column) {

    printf("\nOutput Matrix:\n");
    for (int i = 0; i < row; ++i) {
        for (int j = 0; j < column; ++j) {
            printf("%d  ", result[i][j]);
            if (j == column - 1)
                printf("\n");
        }
    }
}

int main() {
    int first[10][10], second[10][10], result[10][10], r1, c1, r2, c2;
    printf("Enter rows and column for the first matrix: ");
    scanf("%d %d", &r1, &c1);
    printf("Enter rows and column for the second matrix: ");
    scanf("%d %d", &r2, &c2);

    // Taking input until
    // 1st matrix columns is not equal to 2nd matrix row
    while (c1 != r2) {
        printf("Error! Enter rows and columns again.\n");
        printf("Enter rows and columns for the first matrix: ");
        scanf("%d%d", &r1, &c1);
        printf("Enter rows and columns for the second matrix: ");
        scanf("%d%d", &r2, &c2);
    }

```

```

    // get elements of the first matrix
    getMatrixElements(first, r1, c1);

    // get elements of the second matrix
    getMatrixElements(second, r2, c2);

    // multiply two matrices.
    multiplyMatrices(first, second, result, r1, c1, r2, c2);

    // display the result
    display(result, r1, c2);

    return 0;
}
```

Output

```
Enter rows and column for the first matrix: 2
3
Enter rows and column for the second matrix: 3
2

Enter elements:
Enter a11: 2
Enter a12: -3
Enter a13: 4
Enter a21: 53
Enter a22: 3
Enter a23: 5

Enter elements:
Enter a11: 3
Enter a12: 3
Enter a21: 5
Enter a22: 0
Enter a31: -3
Enter a32: 4

Output Matrix:
-21  22
159  179
```

h) C program to multiply 2 Matrix by passing matrix to a function.

```c
#include <stdio.h>

void multiplyMatrices(int A[][10], int B[][10], int C[][10], int r1, int c1, int r2, int c2) {
    if (c1 != r2) {
        printf("Matrices cannot be multiplied.\n");
        return;
    }

    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            C[i][j] = 0;
            for (int k = 0; k < c1; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

int main() {
    int A[10][10], B[10][10], C[10][10], r1, c1, r2, c2;

    printf("Enter the number of rows and columns of matrix A: ");
    scanf("%d %d", &r1, &c1);

    printf("Enter the elements of matrix A:\n");
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c1; j++) {
            scanf("%d", &A[i][j]);
        }
    }

```

```c
    printf("Enter the number of rows and columns of matrix B: ");
    scanf("%d %d", &r2, &c2);

    if (c1 != r2) {
        printf("Matrices cannot be multiplied.\n");
        return 1;
    }

    printf("Enter the elements of matrix B:\n");
    for (int i = 0; i < r2; i++) {
        for (int j = 0; j < c2; j++) {
            scanf("%d", &B[i][j]);
        }
    }

    multiplyMatrices(A, B, C, r1, c1, r2, c2);

    printf("Product of the matrices:\n");
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Microsoft Edge

Output:

```
Enter the number of rows and columns of matrix A: 2
2
Enter the elements of matrix A:
1
2
3
4
Enter the number of rows and columns of matrix B: 2
2
Enter the elements of matrix B:
5
6
7
8
Product of the matrices:
19 22
43 50
```

j) C program to swap numbers in cyclic order using call by reference.

```c
#include <stdio.h>
void cyclicSwap(int *a, int *b, int *c);
int main() {
    int a, b, c;

    printf("Enter a, b and c respectively: ");
    scanf("%d %d %d", &a, &b, &c);

    printf("Value before swapping:\n");
    printf("a = %d \nb = %d \nc = %d\n", a, b, c);

    cyclicSwap(&a, &b, &c);

    printf("Value after swapping:\n");
    printf("a = %d \nb = %d \nc = %d", a, b, c);

    return 0;
}

void cyclicSwap(int *n1, int *n2, int *n3) {
    int temp;
    // swapping in cyclic order
    temp = *n2;
    *n2 = *n1;
    *n1 = *n3;
    *n3 = temp;
}
```

Output

```
Enter a, b and c respectively: 1
2
3
Value before swapping:
a = 1
b = 2
c = 3
Value after swapping:
a = 3
b = 1
c = 2
```

k) C program to find largest number using dynamic memory allocation.

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

int main() {

  int n;
  double *data;
  printf("Enter the total number of elements: ");
  scanf("%d", &n);

  // Allocating memory for n elements
  data = (double *)calloc(n, sizeof(double));
  if (data == NULL) {
    printf("Error!!! memory not allocated.");
    exit(0);
  }

  // Storing numbers entered by the user.
  for (int i = 0; i < n; ++i) {
    printf("Enter number%d: ", i + 1);
    scanf("%lf", data + i);
  }

  // Finding the largest number
  for (int i = 1; i < n; ++i) {
    if (*data < *(data + i)) {
      *data = *(data + i);
    }
  }
  printf("Largest number = %.2lf", *data);

  free(data);

  return 0;
}
```

Output

```
Enter the total number of elements: 5
Enter number1: 3.4
Enter number2: 2.4
Enter number3: -5
Enter number4: 24.2
Enter number5: 6.7
Largest number = 24.20
```

5. String

b) C program to count the number of vowels, consonants and so on.

```c
#include <ctype.h>
#include <stdio.h>
int main() {

  char line[150];
  int vowels, consonant, digit, space;

  // initialize all variables to 0
  vowels = consonant = digit = space = 0;

  // get full line of string input
  printf("Enter a line of string: ");
  fgets(line, sizeof(line), stdin);

  // loop through each character of the string
  for (int i = 0; line[i] != '\0'; ++i) {

    // convert character to lowercase
    line[i] = tolower(line[i]);

    // check if the character is a vowel
    if (line[i] == 'a' || line[i] == 'e' || line[i] == 'i' ||
        line[i] == 'o' || line[i] == 'u') {

      // increment value of vowels by 1
      ++vowels;
    }

    // if it is not a vowel and if it is an alphabet, it is a consonant
    else if ((line[i] >= 'a' && line[i] <= 'z')) {
      ++consonant;
    }

    // check if the character is a digit
    else if (line[i] >= '0' && line[i] <= '9') {
      ++digit;
    }

    // check if the character is an empty space
    else if (line[i] == ' ') {
      ++space;
    }
  }

  printf("Vowels: %d", vowels);
  printf("\nConsonants: %d", consonant);
  printf("\nDigits: %d", digit);
  printf("\nWhite spaces: %d", space);

  return 0;
}
```

Output

```
Enter a line of string: C++ 20 is the latest version of C++ yet.
Vowels: 9
Consonants: 16
Digits: 2
White spaces: 8
```

c) C program to remove all the characters in a string except alphabet.

```c
#include <stdio.h>
int main() {
   char line[150];

   printf("Enter a string: ");
   fgets(line, sizeof(line), stdin); // take input


   for (int i = 0, j; line[i] != '\0'; ++i) {

      // enter the loop if the character is not an alphabet
      // and not the null character
      while (!(line[i] >= 'a' && line[i] <= 'z') && !(line[i] >= 'A' && line[i] <= 'Z') && !(line[i] == '\0')) {
         for (j = i; line[j] != '\0'; ++j) {

            // if jth element of line is not an alphabet,
            // assign the value of (j+1)th element to the jth element
            line[j] = line[j + 1];
         }
         line[j] = '\0';
      }
   }
   printf("Output String: ");
   puts(line);
   return 0;
}
```

Output

```
Enter a string: p2'r-o@gram84iz./
Output String: programiz
```

g) C program to sort elements in lexicographical order (Dictionary order)

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

int main() {
   char str[5][50], temp[50];
   printf("Enter 5 words: ");

   // Getting strings input
   for (int i = 0; i < 5; ++i) {
      fgets(str[i], sizeof(str[i]), stdin);
   }

   // storing strings in the lexicographical order
   for (int i = 0; i < 5; ++i) {
      for (int j = i + 1; j < 5; ++j) {

         // swapping strings if they are not in the lexicographical order
         if (strcmp(str[i], str[j]) > 0) {
            strcpy(temp, str[i]);
            strcpy(str[i], str[j]);
            strcpy(str[j], temp);
         }
      }
   }

   printf("\nIn the lexicographical order: \n");
   for (int i = 0; i < 5; ++i) {
      fputs(str[i], stdout);
   }
   return 0;
}
```

Output

```
Enter 5 words: R programming
JavaScript
Java
C programming
C++ programming

In the lexicographical order:
C programming
C++ programming
Java
JavaScript
R programming
```

6. Structure and union

c) C program to add two complex numbers by passing structure to a function.

```c
#include <stdio.h>
typedef struct complex {
    float real;
    float imag;
} complex;

complex add(complex n1, complex n2);

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

    printf("For 1st complex number \n");
    printf("Enter the real and imaginary parts: ");
    scanf("%f %f", &n1.real, &n1.imag);
    printf("\nFor 2nd complex number \n");
    printf("Enter the real and imaginary parts: ");
    scanf("%f %f", &n2.real, &n2.imag);

    result = add(n1, n2);

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

complex add(complex n1, complex n2) {
    complex temp;
    temp.real = n1.real + n2.real;
    temp.imag = n1.imag + n2.imag;
    return (temp);
}
```

Output

```
For 1st complex number
Enter the real and imaginary parts: 2.1
-2.3

For 2nd complex number
Enter the real and imaginary parts: 5.6
23.2
Sum = 7.7 + 20.9i
```

d) C program to calculate difference between two time periods

```c
#include <stdio.h>

struct Time {
    int hours;
    int minutes;
    int seconds;
};
struct Time timeDifference(struct Time t1, struct Time t2) {
    struct Time diff;
    if (t2.seconds > t1.seconds) {
        --t1.minutes;
        t1.seconds += 60;
    }

    diff.seconds = t1.seconds - t2.seconds;

    if (t2.minutes > t1.minutes) {
        --t1.hours;
        t1.minutes += 60;
    }
    diff.minutes = t1.minutes - t2.minutes;
    diff.hours = t1.hours - t2.hours;
    return diff;
}
int main() {
    struct Time startTime, endTime, diff;

    printf("Enter start time (hh:mm:ss): ");
    scanf("%d:%d:%d", &startTime.hours, &startTime.minutes, &startTime.seconds);
    printf("Enter end time (hh:mm:ss): ");
    scanf("%d:%d:%d", &endTime.hours, &endTime.minutes, &endTime.seconds);

    diff = timeDifference(startTime, endTime);
    printf("\nTime Difference: %d:%d:%d\n", diff.hours, diff.minutes, diff.seconds);
    return 0;
}
```

Output:

```
Enter start time (hh:mm:ss): 12:00:00
Enter end time (hh:mm:ss): 18:00:00

Time Difference: -6:0:0
```

Q. C program to print various geometric patterns using nested loops

```c
//1. Right-angled triangle:
#include <stdio.h>

int main() {
    int rows, i, j;

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for (i = 1; i <= rows; ++i) {
        for (j = 1; j <= i; ++j) {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

```
Enter the number of rows: 5
*
* *
* * *
* * * *
* * * * *
```

```c
//2.Inverted right-angled triangle:
#include <stdio.h>

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

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for (i = rows; i >= 1; --i) {
        for (space = 1; space <= rows - i; ++space) {
            printf("  ");
        }
        for (j = 1; j <= i; ++j) {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

```
Enter the number of rows: 5
* * * * *
  * * * *
    * * *
      * *
        *
```

```c
//3. Pyramid:
#include <stdio.h>

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

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for (i = 1; i <= rows; ++i) {
        for (space = 1; space <= rows - i; ++space) {
            printf("  ");
        }
        for (j = 1; j <= 2 * i - 1; ++j) {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

```
Enter the number of rows: 5
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
```

```c
//4. Diamond:
#include <stdio.h>

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

    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for (i = 1; i <= rows; ++i) {
        for (space = 1; space <= rows - i; ++space) {
            printf("  ");
        }
        for (j = 1; j <= 2 * i - 1; ++j) {
            printf("* ");
        }
        printf("\n");
    }

    for (i = rows - 1; i >= 1; --i) {
        for (space = 1; space <= rows - i; ++space) {
            printf("  ");
        }
        for (j = 1; j <= 2 * i - 1; ++j) {
            printf("* ");
        }
        printf("\n");
    }

    return 0;
}
```

Output:

```
Enter the number of rows: 5
        *
      * * *
    * * * * *
  * * * * * * *
* * * * * * * * *
  * * * * * * *
    * * * * *
      * * *
        *
```

Q. C Program to find if substring exist in a string or not

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

int main() {
    char str[100], substr[100];

    printf("Enter the string: ");
    fgets(str, 100, stdin);

    printf("Enter the substring to search: ");
    fgets(substr, 100, stdin);

    // Remove trailing newline characters
    str[strcspn(str, "\n")] = '\0';
    substr[strcspn(substr, "\n")] = '\0';

    char *result = strstr(str, substr);

    if (result != NULL) {
        printf("Substring found!\n");
    } else {
        printf("Substring not found.\n");
    }

    return 0;
}
```

```
Enter the string: hello
Enter the substring to search: he
Substring found!
```