Sri Lanka Institute of Information Technology

IT21299902 (ZAKEY M.S.M.A)

FA assignment report.

Foundation of Algorithms – IE2072

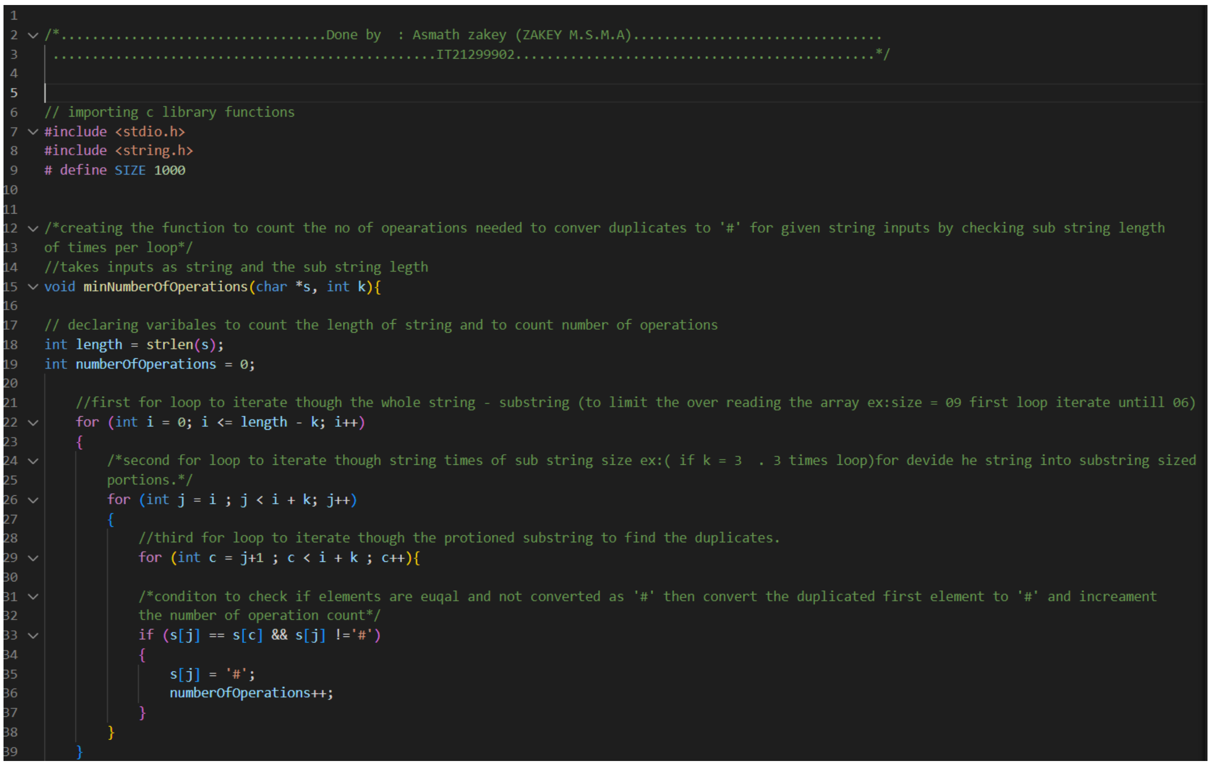
B.Sc. (Hons) in Information Technology Specialization in cyber security.

Question 1 – Source code(snapshot)

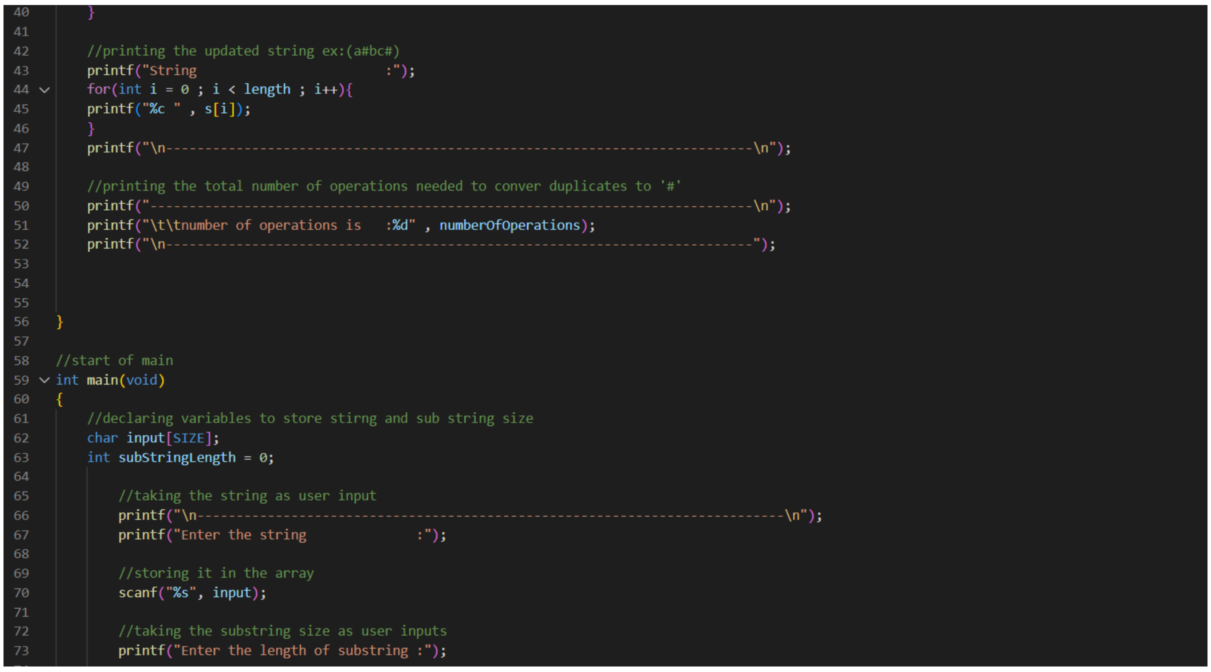
You are given a string S containing only lowercase letters and an integer K. In one operation you can change any character of the string to '#' character.

Note: '#' is not considered when checking for duplicates.

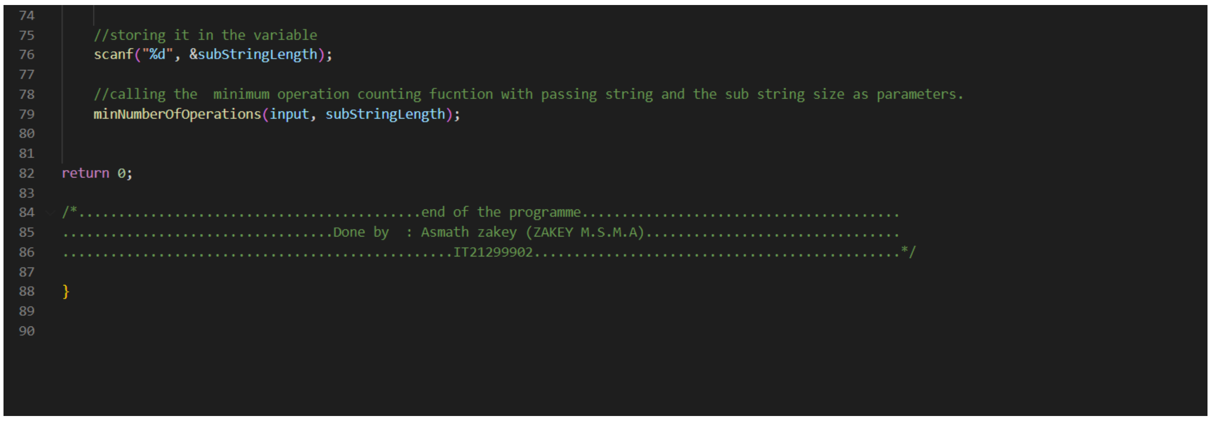
Print the minimum number of operations required such that no substring of size K contains duplicates.



```c

/*................................Done by  : Asmath zakey (ZAKEY M.S.M.A)...............................
 ................................................IT21299902.............................................*/


// importing c library functions
#include <stdio.h>
#include <string.h>
# define SIZE 1000


/*creating the function to count the no of opearations needed to conver duplicates to '#' for given string inputs by checking sub string length
of times per loop*/
//takes inputs as string and the sub string legth
void minNumberOfOperations(char *s, int k){

// declaring varibales to count the length of string and to count number of operations
int length = strlen(s);
int numberOfOperations = 0;

    //first for loop to iterate though the whole string - substring (to limit the over reading the array ex:size = 09 first loop iterate untill 06)
    for (int i = 0; i <= length - k; i++)
    {
        /*second for loop to iterate though string times of sub string size ex:( if k = 3  . 3 times loop)for devide he string into substring sized
        portions.*/
        for (int j = i ; j < i + k; j++)
        {
            //third for loop to iterate though the protioned substring to find the duplicates.
            for (int c = j+1 ; c < i + k ; c++){

            /*conditon to check if elements are euqal and not converted as '#' then convert the duplicated first element to '#' and increament
            the number of operation count*/
            if (s[j] == s[c] && s[j] !='#')
            {
                s[j] = '#';
                numberOfOperations++;
            }
        }
    }
```

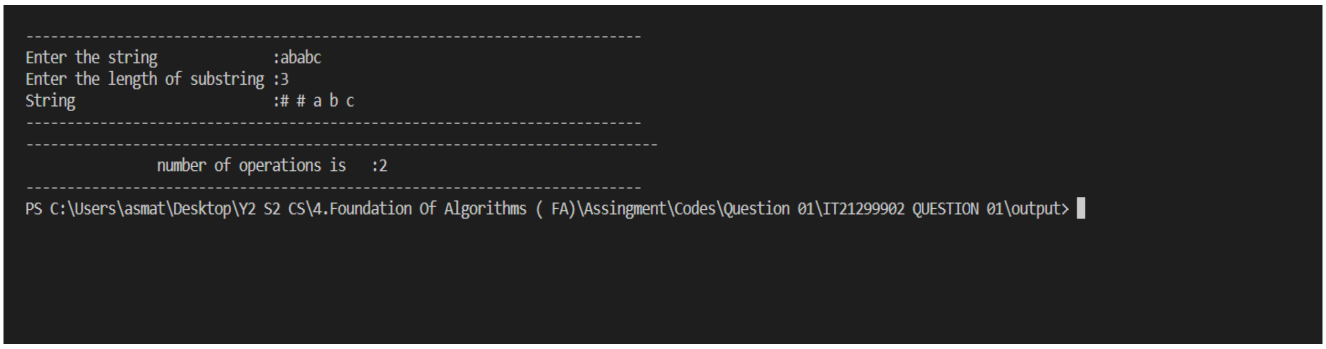


```c
    }

    //printing the updated string ex:(a#bc#)
    printf("String                        :");
    for(int i = 0 ; i < length ; i++){
    printf("%c " , s[i]);
    }
    printf("\n--------------------------------------------------------------------------\n");

    //printing the total number of operations needed to conver duplicates to '#'
    printf("----------------------------------------------------------------------------\n");
    printf("\t\tnumber of operations is   :%d" , numberOfOperations);
    printf("\n--------------------------------------------------------------------------");



}

//start of main
int main(void)
{
    //declaring variables to store stirng and sub string size
    char input[SIZE];
    int subStringLength = 0;

        //taking the string as user input
        printf("\n--------------------------------------------------------------------------\n");
        printf("Enter the string               :");

        //storing it in the array
        scanf("%s", input);

        //taking the substring size as user inputs
        printf("Enter the length of substring :");
```

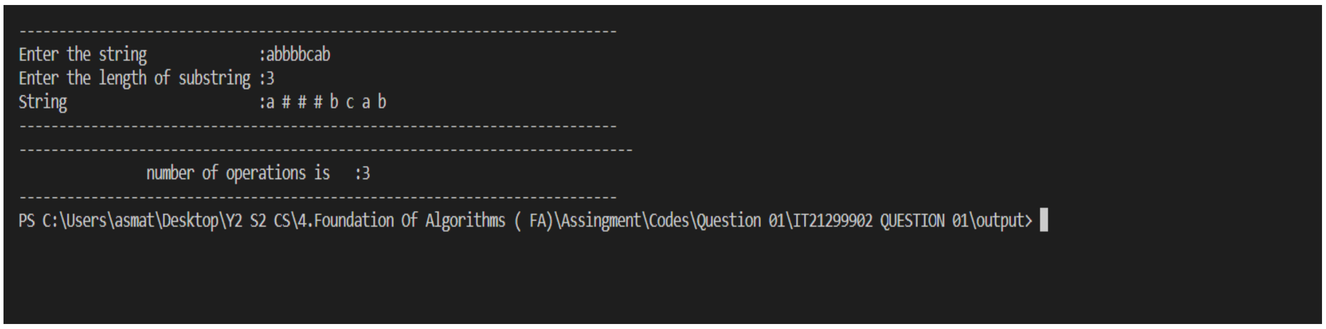


```
        //storing it in the variable
        scanf("%d", &subStringLength);

        //calling the  minimum operation counting fucntion with passing string and the sub string size as parameters.
        minNumberOfOperations(input, subStringLength);


    return 0;

    /*.........................................end of the programme......................................
    ................................Done by  : Asmath zakey (ZAKEY M.S.M.A)..............................
    ..............................................IT21299902.............................................*/

    }
```

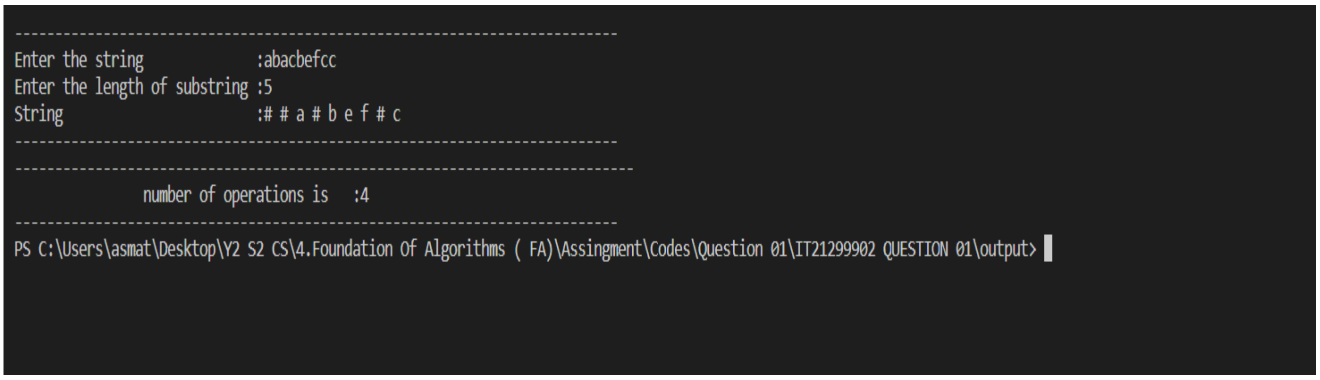
Question 1 – Outputs



```
---------------------------------------------------------------------------
Enter the string              :ababc
Enter the length of substring :3
String                        :# # a b c
---------------------------------------------------------------------------
-----------------------------------------------------------------------------
                number of operations is   :2
---------------------------------------------------------------------------
PS C:\Users\asmat\Desktop\Y2 S2 CS\4.Foundation Of Algorithms ( FA)\Assingment\Codes\Question 01\IT21299902 QUESTION 01\output>
```



```
---------------------------------------------------------------------------
Enter the string              :abbbbcab
Enter the length of substring :3
String                        :a # # # b c a b
---------------------------------------------------------------------------
-----------------------------------------------------------------------------
                number of operations is   :3
---------------------------------------------------------------------------
PS C:\Users\asmat\Desktop\Y2 S2 CS\4.Foundation Of Algorithms ( FA)\Assingment\Codes\Question 01\IT21299902 QUESTION 01\output>
```



```
---------------------------------------------------------------------------
Enter the string              :abacbefcc
Enter the length of substring :5
String                        :# # a # b e f # c
---------------------------------------------------------------------------
-----------------------------------------------------------------------------
                number of operations is   :4
---------------------------------------------------------------------------
PS C:\Users\asmat\Desktop\Y2 S2 CS\4.Foundation Of Algorithms ( FA)\Assingment\Codes\Question 01\IT21299902 QUESTION 01\output>
```

Source code question 01

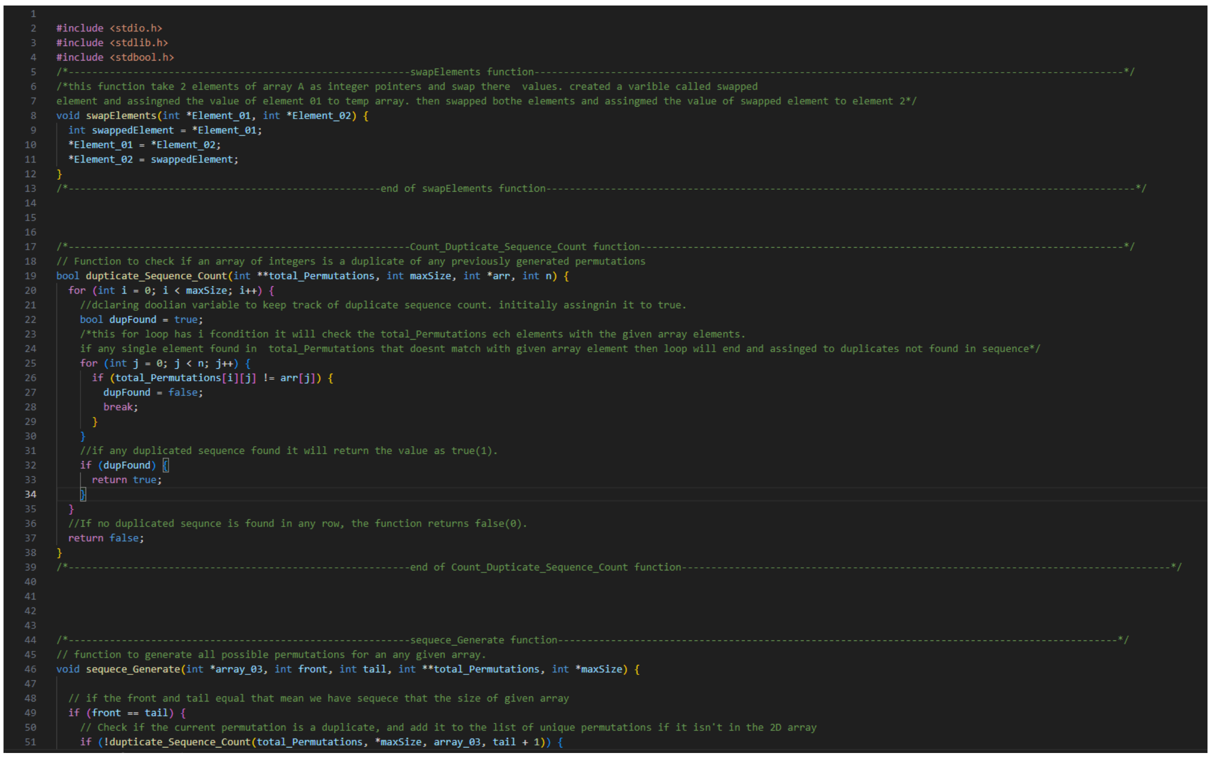
```
/*.................................Done by  : Asmath zakey (ZAKEY
M.S.M.A)...............................
 .................................................IT21299902..................
............................*/


// importing c library functions
#include <stdio.h>
#include <string.h>
# define SIZE 1000


/*creating the function to count the no of opearations needed to conver
duplicates to '#' for given string inputs by checking sub string length
of times per loop*/
//takes inputs as string and the sub string legth
void minNumberOfOperations(char *s, int k){

// declaring varibales to count the length of string and to count number of
operations
int length = strlen(s);
int numberOfOperations = 0;

    //first for loop to iterate though the whole string - substring (to limit
the over reading the array ex:size = 09 first loop iterate untill 06)
    for (int i = 0; i <= length - k; i++)
    {
        /*second for loop to iterate though string times of sub string size
ex:( if k = 3  . 3 times loop)for devide he string into substring sized
        portions.*/
        for (int j = i ; j < i + k; j++)
        {
            //third for loop to iterate though the protioned substring to find
the duplicates.
            for (int c = j+1 ; c < i + k ; c++){

            /*conditon to check if elements are euqal and not converted as '#'
then convert the duplicated first element to '#' and increament
            the number of operation count*/
            if (s[j] == s[c] && s[j] !='#')
            {
                s[j] = '#';
                numberOfOperations++;
            }
        }
    }
    }

    //printing the updated string ex:(a#bc#)
    printf("String                        :");
    for(int i = 0 ; i < length ; i++){
    printf("%c " , s[i]);
    }
    printf("\n----------------------------------------------------------------
-----------\n");

    //printing the total number of operations needed to conver duplicates to
'#'
    printf("------------------------------------------------------------------
-----------\n");
    printf("\t\tnumber of operations is   :%d" , numberOfOperations);
    printf("\n----------------------------------------------------------------
-----------");



}

//start of main
int main(void)
{
    //declaring variables to store stirng and sub string size
    char input[SIZE];
    int subStringLength = 0;

        //taking the string as user input
        printf("\n------------------------------------------------------------
---------------\n");
        printf("Enter the string              :");

        //storing it in the array
        scanf("%s", input);

        //taking the substring size as user inputs
        printf("Enter the length of substring :");

    //storing it in the variable
    scanf("%d", &subStringLength);

    //calling the  minimum operation counting fucntion with passing string and
the sub string size as parameters.
    minNumberOfOperations(input, subStringLength);
```

```
return 0;

/*...........................................end of the
programme........................................
..................................Done by  : Asmath zakey (ZAKEY
M.S.M.A)................................
.................................................IT21299902...................
...........................*/

}
```

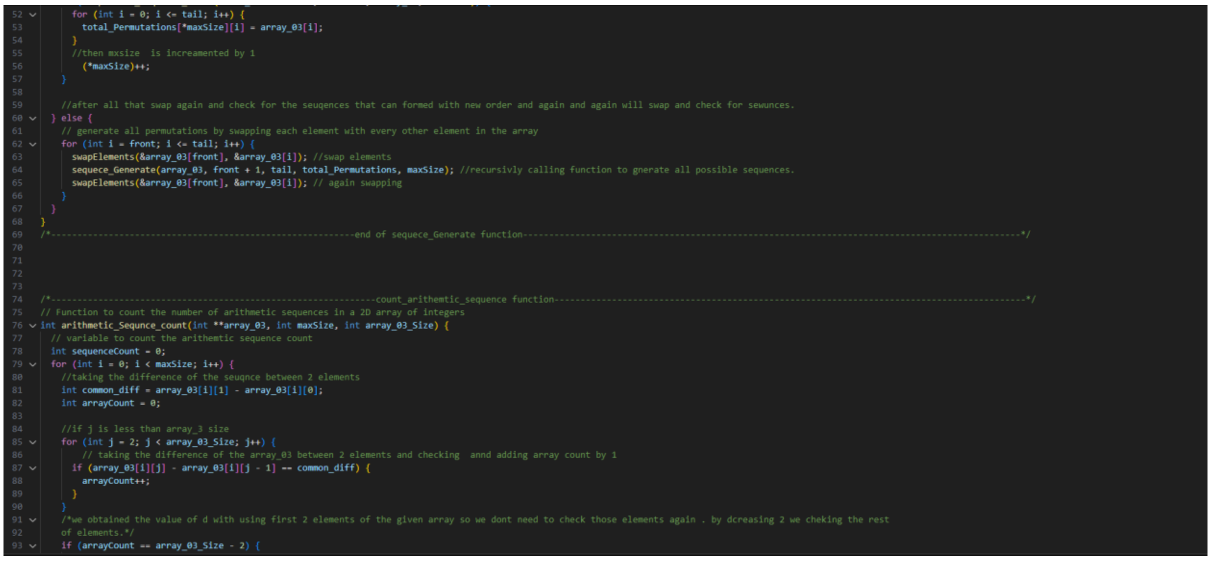
Question 2 – Source code(snapshot)

You are given two non-decreasing sequences A=(a1,a2,...,an) and B=(b1,b2,...,bm). You can choose any two indices i and j, and then swap ai and bj.

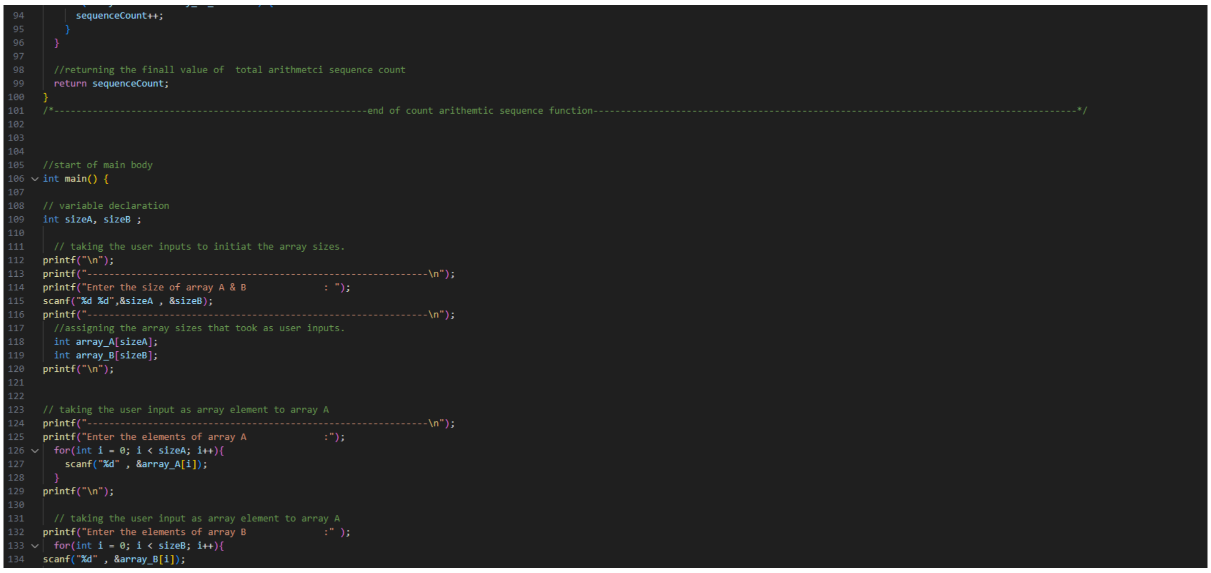
Note that you can do the operations as many times as you want, and your goal is to transform A into an arithmetic sequence. After operations, you can rearrange the sequence A. Determine how many distinct arithmetic sequences you can obtain.



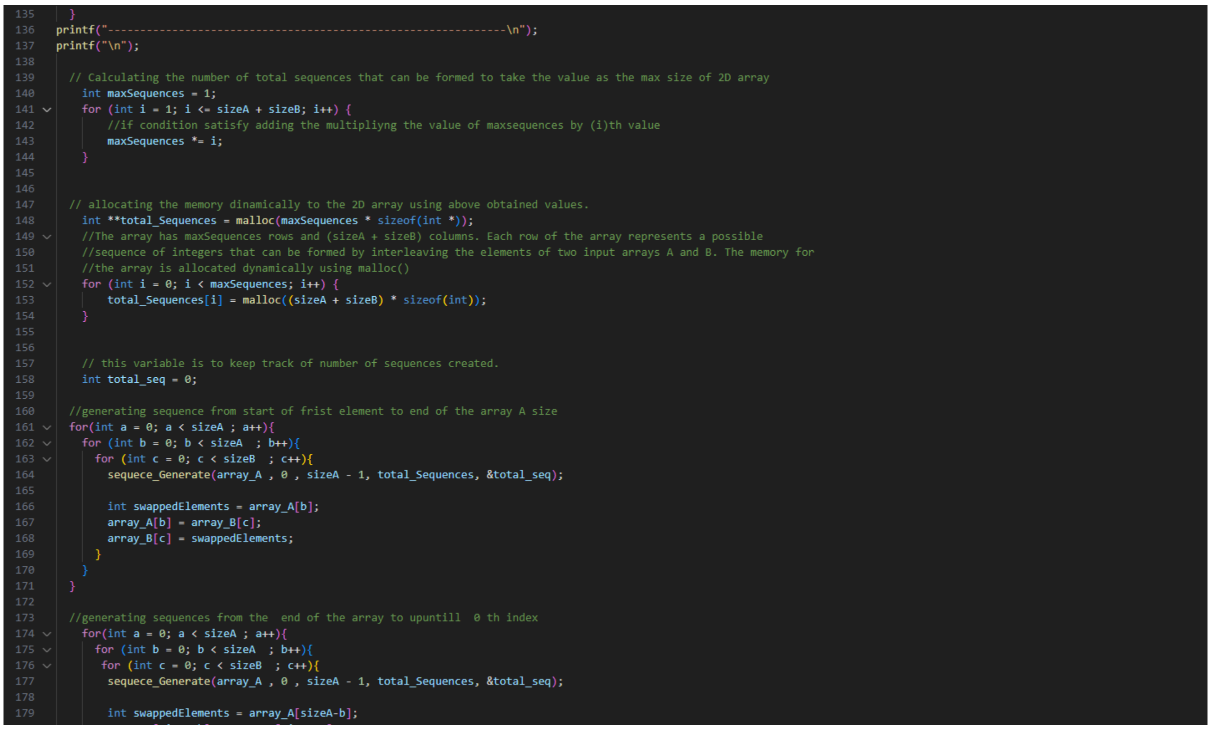
```c

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
/*-----------------------------------------------------------swapElements function-------------------------------------------------------------------------------------------------*/
/*this function take 2 elements of array A as integer pointers and swap there  values. created a varible called swapped
element and assingned the value of element 01 to temp array. then swapped bothe elements and assingmed the value of swapped element to element 2*/
void swapElements(int *Element_01, int *Element_02) {
  int swappedElement = *Element_01;
  *Element_01 = *Element_02;
  *Element_02 = swappedElement;
}
/*------------------------------------------------------end of swapElements function------------------------------------------------------------------------------------------------------*/



/*-----------------------------------------------------------Count_Dupticate_Sequence_Count function----------------------------------------------------------------------------------*/
// Function to check if an array of integers is a duplicate of any previously generated permutations
bool dupticate_Sequence_Count(int **total_Permutations, int maxSize, int *arr, int n) {
  for (int i = 0; i < maxSize; i++) {
    //dclaring doolian variable to keep track of duplicate sequence count. inititally assingnin it to true.
    bool dupFound = true;
    /*this for loop has i fcondition it will check the total_Permutations ech elements with the given array elements.
    if any single element found in  total_Permutations that doesnt match with given array element then loop will end and assinged to duplicates not found in sequence*/
    for (int j = 0; j < n; j++) {
      if (total_Permutations[i][j] != arr[j]) {
        dupFound = false;
        break;
      }
    }
    //if any duplicated sequence found it will return the value as true(1).
    if (dupFound) {
      return true;
    }
  }
  //If no duplicated sequnce is found in any row, the function returns false(0).
  return false;
}
/*-----------------------------------------------------------end of Count_Dupticate_Sequence_Count function-----------------------------------------------------------------------------*/




/*-----------------------------------------------------------sequece_Generate function-----------------------------------------------------------------------------------------------*/
// function to generate all possible permutations for an any given array.
void sequece_Generate(int *array_03, int front, int tail, int **total_Permutations, int *maxSize) {

  // if the front and tail equal that mean we have sequece that the size of given array
  if (front == tail) {
    // Check if the current permutation is a duplicate, and add it to the list of unique permutations if it isn't in the 2D array
    if (!dupticate_Sequence_Count(total_Permutations, *maxSize, array_03, tail + 1)) {
```



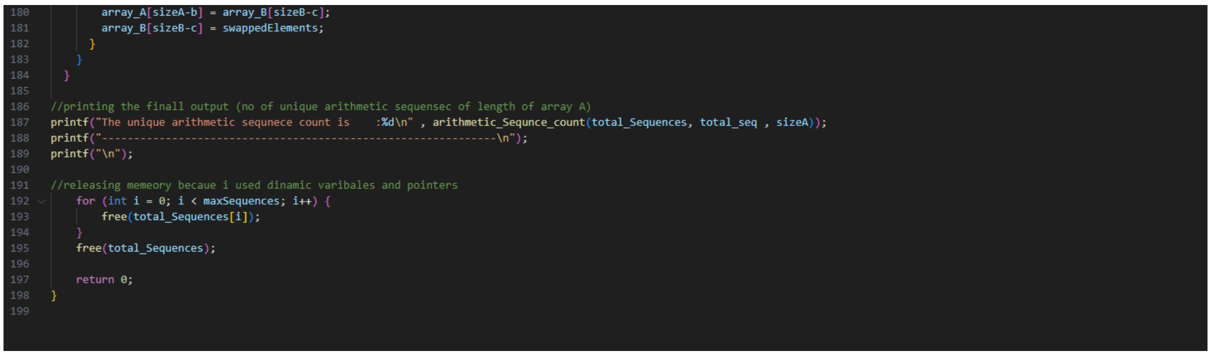
```c
      for (int i = 0; i <= tail; i++) {
        total_Permutations[*maxSize][i] = array_03[i];
      }
      //then mxsize  is increamented by 1
        (*maxSize)++;
    }

    //after all that swap again and check for the seuqences that can formed with new order and again and again will swap and check for sewunces.
  } else {
    // generate all permutations by swapping each element with every other element in the array
    for (int i = front; i <= tail; i++) {
      swapElements(&array_03[front], &array_03[i]); //swap elements
      sequece_Generate(array_03, front + 1, tail, total_Permutations, maxSize); //recursivly calling function to gnerate all possible sequences.
      swapElements(&array_03[front], &array_03[i]); // again swapping
    }
  }
}
/*-----------------------------------------------------------end of sequece_Generate function-----------------------------------------------------------------------------------*/




/*-----------------------------------------------------------count_arithemtic_sequence function---------------------------------------------------------------------------------*/
// Function to count the number of arithmetic sequences in a 2D array of integers
int arithmetic_Sequnce_count(int **array_03, int maxSize, int array_03_Size) {
  // variable to count the arithemtic sequence count
  int sequenceCount = 0;
  for (int i = 0; i < maxSize; i++) {
    //taking the difference of the seuqnce between 2 elements
    int common_diff = array_03[i][1] - array_03[i][0];
    int arrayCount = 0;

    //if j is less than array_3 size
    for (int j = 2; j < array_03_Size; j++) {
        // taking the difference of the array_03 between 2 elements and checking  annd adding array count by 1
      if (array_03[i][j] - array_03[i][j - 1] == common_diff) {
        arrayCount++;
      }
    }
    /*we obtained the value of d with using first 2 elements of the given array so we dont need to check those elements again . by dcreasing 2 we cheking the rest
    of elements.*/
    if (arrayCount == array_03_Size - 2) {
```



```c
            sequenceCount++;
        }
    }

    //returning the finall value of  total arithmetci sequence count
    return sequenceCount;
}
/*--------------------------------------------------------end of count arithemtic sequence function-------------------------------------------------------------------------------------*/



//start of main body
int main() {

// variable declaration
int sizeA, sizeB ;

  // taking the user inputs to initiat the array sizes.
printf("\n");
printf("------------------------------------------------------------\n");
printf("Enter the size of array A & B              : ");
scanf("%d %d",&sizeA , &sizeB);
printf("------------------------------------------------------------\n");
  //assigning the array sizes that took as user inputs.
  int array_A[sizeA];
  int array_B[sizeB];
printf("\n");


// taking the user input as array element to array A
printf("------------------------------------------------------------\n");
printf("Enter the elements of array A              :");
  for(int i = 0; i < sizeA; i++){
    scanf("%d" , &array_A[i]);
  }
printf("\n");

  // taking the user input as array element to array A
printf("Enter the elements of array B              :" );
  for(int i = 0; i < sizeB; i++){
scanf("%d" , &array_B[i]);
```

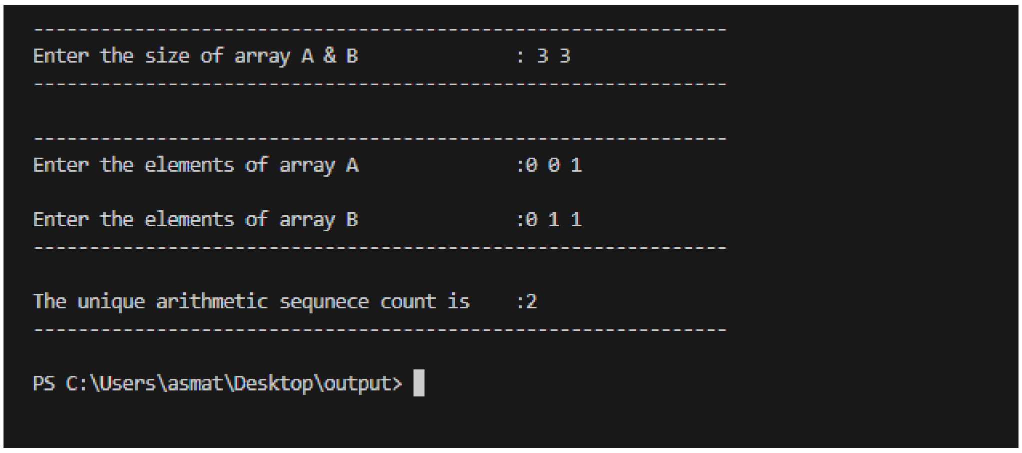


```c
  }
printf("------------------------------------------------------------\n");
printf("\n");

  // Calculating the number of total sequences that can be formed to take the value as the max size of 2D array
    int maxSequences = 1;
    for (int i = 1; i <= sizeA + sizeB; i++) {
        //if condition satisfy adding the multipliyng the value of maxsequences by (i)th value
        maxSequences *= i;
    }


  // allocating the memory dinamically to the 2D array using above obtained values.
    int **total_Sequences = malloc(maxSequences * sizeof(int *));
    //The array has maxSequences rows and (sizeA + sizeB) columns. Each row of the array represents a possible
    //sequence of integers that can be formed by interleaving the elements of two input arrays A and B. The memory for
    //the array is allocated dynamically using malloc()
    for (int i = 0; i < maxSequences; i++) {
        total_Sequences[i] = malloc((sizeA + sizeB) * sizeof(int));
    }


    // this variable is to keep track of number of sequences created.
    int total_seq = 0;

  //generating sequence from start of frist element to end of the array A size
  for(int a = 0; a < sizeA ; a++){
    for (int b = 0; b < sizeA  ; b++){
      for (int c = 0; c < sizeB  ; c++){
        sequece_Generate(array_A , 0 , sizeA - 1, total_Sequences, &total_seq);

        int swappedElements = array_A[b];
        array_A[b] = array_B[c];
        array_B[c] = swappedElements;
      }
    }
  }

  //generating sequences from the  end of the array to upuntill  0 th index
    for(int a = 0; a < sizeA ; a++){
      for (int b = 0; b < sizeA  ; b++){
       for (int c = 0; c < sizeB  ; c++){
        sequece_Generate(array_A , 0 , sizeA - 1, total_Sequences, &total_seq);

        int swappedElements = array_A[sizeA-b];
```

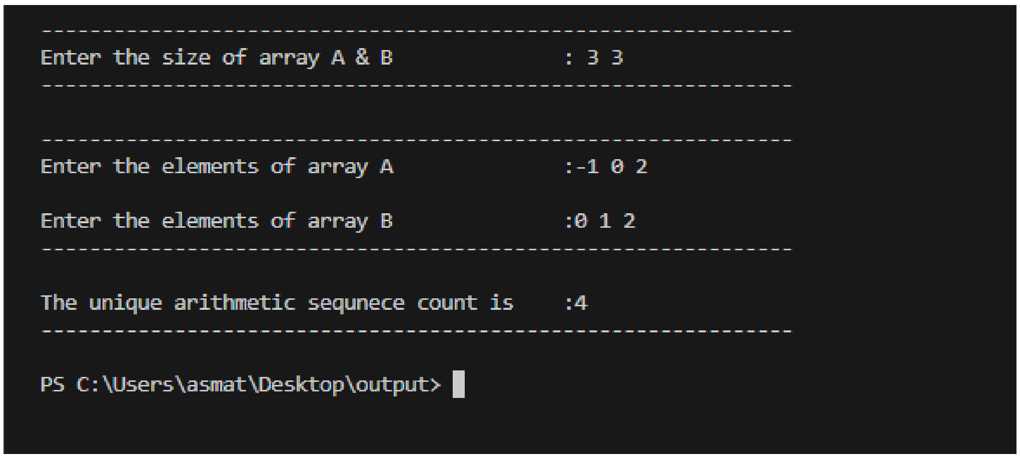


```c
        array_A[sizeA-b] = array_B[sizeB-c];
        array_B[sizeB-c] = swappedElements;
      }
    }
  }

  //printing the finall output (no of unique arithmetic sequensec of length of array A)
  printf("The unique arithmetic sequnece count is    :%d\n" , arithmetic_Sequnce_count(total_Sequences, total_seq , sizeA));
  printf("------------------------------------------------------------\n");
  printf("\n");

  //releasing memeory becaue i used dinamic varibales and pointers
      for (int i = 0; i < maxSequences; i++) {
          free(total_Sequences[i]);
      }
      free(total_Sequences);

      return 0;
  }

```

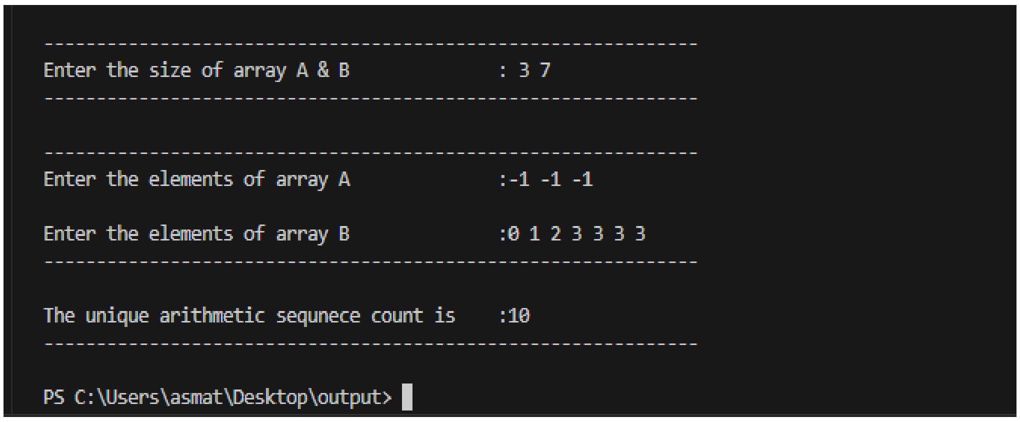
Question 2 – Sample Output



```
--------------------------------------------------------------
Enter the size of array A & B              : 3 3
--------------------------------------------------------------

--------------------------------------------------------------
Enter the elements of array A              :0 0 1

Enter the elements of array B              :0 1 1
--------------------------------------------------------------

The unique arithmetic sequnece count is    :2
--------------------------------------------------------------

PS C:\Users\asmat\Desktop\output>
```



```
--------------------------------------------------------------
Enter the size of array A & B              : 3 3
--------------------------------------------------------------

--------------------------------------------------------------
Enter the elements of array A              :-1 0 2

Enter the elements of array B              :0 1 2
--------------------------------------------------------------

The unique arithmetic sequnece count is    :4
--------------------------------------------------------------

PS C:\Users\asmat\Desktop\output>
```



```
--------------------------------------------------------------
Enter the size of array A & B              : 3 7
--------------------------------------------------------------

--------------------------------------------------------------
Enter the elements of array A              :-1 -1 -1

Enter the elements of array B              :0 1 2 3 3 3 3
--------------------------------------------------------------

The unique arithmetic sequnece count is    :10
--------------------------------------------------------------

PS C:\Users\asmat\Desktop\output>
```

Source code question 02

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
/*----------------------------------------------------------swapElements
function------------------------------------------------------------------------------
------------------------------*/
/*this function take 2 elements of array A as integer pointers and swap
there  values. created a varible called swapped
element and assingned the value of element 01 to temp array. then swapped
bothe elements and assingmed the value of swapped element to element 2*/
void swapElements(int *Element_01, int *Element_02) {
  int swappedElement = *Element_01;
  *Element_01 = *Element_02;
  *Element_02 = swappedElement;
}
/*-----------------------------------------------------end of swapElements
function------------------------------------------------------------------------------
------------------------------*/



/*----------------------------------------------------------
Count_Dupticate_Sequence_Count function---------------------------------------
-------------------------------------------*/
// Function to check if an array of integers is a duplicate of any previously
generated permutations
bool dupticate_Sequence_Count(int **total_Permutations, int maxSize, int *arr,
int n) {
  for (int i = 0; i < maxSize; i++) {
    //dclaring doolian variable to keep track of duplicate sequence count.
inititally assingnin it to true.
    bool dupFound = true;
    /*this for loop has i fcondition it will check the total_Permutations ech
elements with the given array elements.
    if any single element found in  total_Permutations that doesnt match with
given array element then loop will end and assinged to duplicates not found in
sequence*/
    for (int j = 0; j < n; j++) {
      if (total_Permutations[i][j] != arr[j]) {
        dupFound = false;
        break;
      }
    }
    //if any duplicated sequence found it will return the value as true(1).
    if (dupFound) {
      return true;
    }
  }
  //If no duplicated sequnce is found in any row, the function returns
false(0).
  return false;
}
/*----------------------------------------------------------end of
Count_Dupticate_Sequence_Count function---------------------------------------
--------------------------------------------*/




/*----------------------------------------------------------sequece_Generate
function----------------------------------------------------------------------
-------------------------*/
// function to generate all possible permutations for an any given array.
void sequece_Generate(int *array_03, int front, int tail, int
**total_Permutations, int *maxSize) {

  // if the front and tail equal that mean we have sequece that the size of
given array
  if (front == tail) {
    // Check if the current permutation is a duplicate, and add it to the list
of unique permutations if it isn't in the 2D array
    if (!dupticate_Sequence_Count(total_Permutations, *maxSize, array_03, tail
+ 1)) {
      for (int i = 0; i <= tail; i++) {
        total_Permutations[*maxSize][i] = array_03[i];
      }
      //then mxsize  is increamented by 1
        (*maxSize)++;
    }

    //after all that swap again and check for the seuqences that can formed
with new order and again and again will swap and check for sewunces.
  } else {
    // generate all permutations by swapping each element with every other
element in the array
    for (int i = front; i <= tail; i++) {
      swapElements(&array_03[front], &array_03[i]); //swap elements
      sequece_Generate(array_03, front + 1, tail, total_Permutations,
maxSize); //recursivly calling function to gnerate all possible sequences.
      swapElements(&array_03[front], &array_03[i]); // again swapping
    }
  }
}
```

```
/*-----------------------------------------------------------end of
sequece_Generate function-----------------------------------------------------------
------------------------------------------*/




/*----------------------------------------------------------------
count_arithemtic_sequence function-------------------------------------------------
----------------------------------------------*/
// Function to count the number of arithmetic sequences in a 2D array of
integers
int arithmetic_Sequnce_count(int **array_03, int maxSize, int array_03_Size) {
  // variable to count the arithemtic sequence count
  int sequenceCount = 0;
  for (int i = 0; i < maxSize; i++) {
    //taking the difference of the seuqnce between 2 elements
    int common_diff = array_03[i][1] - array_03[i][0];
    int arrayCount = 0;

    //if j is less than array_3 size
    for (int j = 2; j < array_03_Size; j++) {
        // taking the difference of the array_03 between 2 elements and
checking  annd adding array count by 1
      if (array_03[i][j] - array_03[i][j - 1] == common_diff) {
        arrayCount++;
      }
    }
    /*we obtained the value of d with using first 2 elements of the given
array so we dont need to check those elements again . by dcreasing 2 we
cheking the rest
    of elements.*/
    if (arrayCount == array_03_Size - 2) {
      sequenceCount++;
    }
  }

  //returning the finall value of  total arithmetci sequence count
  return sequenceCount;
}
/*----------------------------------------------------------end of count
arithemtic sequence function-------------------------------------------------------
----------------------------------------*/




//start of main body
int main() {
```

```c
// variable declaration
int sizeA, sizeB ;

  // taking the user inputs to initiat the array sizes.
printf("\n");
printf("--------------------------------------------------------------\n");
printf("Enter the size of array A & B              : ");
scanf("%d %d",&sizeA , &sizeB);
printf("--------------------------------------------------------------\n");
  //assigning the array sizes that took as user inputs.
  int array_A[sizeA];
  int array_B[sizeB];
printf("\n");


// taking the user input as array element to array A
printf("--------------------------------------------------------------\n");
printf("Enter the elements of array A             :");
  for(int i = 0; i < sizeA; i++){
    scanf("%d" , &array_A[i]);
  }
printf("\n");

  // taking the user input as array element to array A
printf("Enter the elements of array B             :" );
  for(int i = 0; i < sizeB; i++){
scanf("%d" , &array_B[i]);
  }
printf("--------------------------------------------------------------\n");
printf("\n");

  // Calculating the number of total sequences that can be formed to take the
value as the max size of 2D array
    int maxSequences = 1;
    for (int i = 1; i <= sizeA + sizeB; i++) {
        //if condition satisfy adding the multipliyng the value of
maxsequences by (i)th value
        maxSequences *= i;
    }


  // allocating the memory dinamically to the 2D array using above obtained
values.
    int **total_Sequences = malloc(maxSequences * sizeof(int *));
    //The array has maxSequences rows and (sizeA + sizeB) columns. Each row of
the array represents a possible
    //sequence of integers that can be formed by interleaving the elements of
two input arrays A and B. The memory for
    //the array is allocated dynamically using malloc()
    for (int i = 0; i < maxSequences; i++) {
        total_Sequences[i] = malloc((sizeA + sizeB) * sizeof(int));
    }


    // this variable is to keep track of number of sequences created.
    int total_seq = 0;

  //generating sequence from start of frist element to end of the array A size
  for(int a = 0; a < sizeA ; a++){
    for (int b = 0; b < sizeA  ; b++){
      for (int c = 0; c < sizeB  ; c++){
        sequece_Generate(array_A , 0 , sizeA - 1, total_Sequences,
&total_seq);

        int swappedElements = array_A[b];
        array_A[b] = array_B[c];
        array_B[c] = swappedElements;
      }
    }
  }

  //generating sequences from the  end of the array to upuntill  0 th index
    for(int a = 0; a < sizeA ; a++){
      for (int b = 0; b < sizeA  ; b++){
       for (int c = 0; c < sizeB  ; c++){
        sequece_Generate(array_A , 0 , sizeA - 1, total_Sequences,
&total_seq);

        int swappedElements = array_A[sizeA-b];
        array_A[sizeA-b] = array_B[sizeB-c];
        array_B[sizeB-c] = swappedElements;
      }
    }
  }

//printing the finall output (no of unique arithmetic sequensec of length of
array A)
printf("The unique arithmetic sequnece count is    :%d\n" ,
arithmetic_Sequnce_count(total_Sequences, total_seq , sizeA));
printf("--------------------------------------------------------------\n");
printf("\n");

//releasing memeory becaue i used dinamic varibales and pointers
    for (int i = 0; i < maxSequences; i++) {
        free(total_Sequences[i]);
    }
    free(total_Sequences);
    return 0;
}
```