

# **Sri Lanka Institute of Information Technology**



**IT21299902 (ZAKY 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.

```

1  /* .....Done by : Asmath zakey (ZAKHEY M.S.M.A)..... */
2  /* .....IT21299902.....*/
3
4
5
6  // importing c library functions
7  #include <stdio.h>
8  #include <string.h>
9  # define SIZE 1000
10
11
12 /*creating the function to count the no of opearations needed to conver duplicates to '#' for given string inputs by checking sub string length
13   of times per loop*/
14 //takes inputs as string and the sub string legh
15 void minNumberOfOperations(char *s, int k){
16
17     // declaring variables to count the length of string and to count number of operations
18     int length = strlen(s);
19     int numberOfOperations = 0;
20
21     //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)
22     for (int i = 0; i <= length - k; i++)
23     {
24         /*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
25           portions.*/
26         for (int j = i ; j < i + k; j++)
27         {
28             //third for loop to iterate though the protioned substring to find the duplicates.
29             for (int c = j+1 ; c < i + k ; c++){
30
31                 /*conditon to check if elements are euqal and not converted as '#' then convert the duplicated first element to '#' and increament
32                   the number of operation count*/
33                 if (s[j] == s[c] && s[j] != '#')
34                 {
35                     s[j] = '#';
36                     numberOfOperations++;
37                 }
38             }
39         }
40     }
41
42     //printing the updated string ex:(a#bc#)
43     printf("String          :");
44     for(int i = 0 ; i < length ; i++){
45         printf("%c " , s[i]);
46     }
47     printf("\n-----\n");
48
49     //printing the total number of operations needed to conver duplicates to '#'
50     printf("-----\n");
51     printf("\t\tnumber of operations is    :%d" , numberOfOperations);
52     printf("\n-----");
53
54 }
55
56 //start of main
57
58 int main(void)
59 {
60     //declaring variables to store stirng and sub string size
61     char input[SIZE];
62     int subStringLength = 0;
63
64     //taking the string as user input
65     printf("\n-----\n");
66     printf("Enter the string          :");
67
68     //storing it in the array
69     scanf("%s", input);
70
71     //taking the substring size as user inputs
72     printf("Enter the length of substring :");

```

```
74
75 //storing it in the variable
76 scanf("%d", &subStringLength);
77
78 //calling the minimum operation counting fucntion with passing string and the sub string size as parameters.
79 minNumberOfOperations(input, subStringLength);
80
81
82 return 0;
83
84 /*.....end of the programme.....
85 .....Done by : Asmath zakey (ZAKEY M.S.M.A).....
86 .....IT21299902.....*/
87
88 }
89
90
```

## 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      :abbbcab  
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");
-----\n");

//printing the total number of operations needed to conver duplicates to
'#'
printf("-----\n");
printf("\t\tnumber of operations is   :%d" , numberOfOperations);
printf("\n-----\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");
    -----\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  $A=(a_1,a_2,\dots,a_n)$  and  $B=(b_1,b_2,\dots,b_m)$ . You can choose any two indices  $i$  and  $j$ , and then swap  $a_i$  and  $b_j$ .

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.

```
1
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <stdbool.h>
5 /*-----swapElements function-----*/
6 /*this function take 2 elements of array A as integer pointers and swap there values. created a variable called swapped
7 element and assigned the value of element 01 to temp array. then swapped both elements and assigned the value of swapped element to element 2*/
8 void swapElements(int *Element_01, int *Element_02) {
9     int swappedElement = *Element_01;
10    *Element_01 = *Element_02;
11    *Element_02 = swappedElement;
12 }
13 /*-----end of swapElements function-----*/
14
15
16
17 /*-----Count_Duplicate_Sequence_Count function-----*/
18 // Function to check if an array of integers is a duplicate of any previously generated permutations
19 bool duplicate_Sequence_Count(int **total_Permutations, int maxSize, int *arr, int n) {
20     for (int i = 0; i < maxSize; i++) {
21         //declaring doolian variable to keep track of duplicate sequence count. initially assignin it to true.
22         bool dupFound = true;
23         /*this for loop has i fcondition it will check the total_Permutations ech elements with the given array elements.
24         if any single element found in total_Permutations that doesnt match with given array element then loop will end and assigned to duplicates not found in sequence*/
25         for (int j = 0; j < n; j++) {
26             if (total_Permutations[i][j] != arr[j]) {
27                 dupFound = false;
28                 break;
29             }
30         }
31         //If any duplicated sequence found it will return the value as true(1).
32         if (dupFound) {
33             return true;
34         }
35     }
36     //If no duplicated sequence is found in any row, the function returns false(0).
37     return false;
38 }
39 /*-----end of Count_Duplicate_Sequence_Count function-----*/
40
41
42
43 /*-----sequence_Generate function-----*/
44 // function to generate all possible permutations for an any given array.
45 void sequence_Generate(int *array_03, int front, int tail, int **total_Permutations, int *maxSize) {
46
47     // if the front and tail equal that mean we have sequece that the size of given array
48     if (front == tail) {
49         // 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
50         if (!duplicate_Sequence_Count(total_Permutations, *maxSize, array_03, tail + 1)) {
51
52             for (int i = 0; i <= tail; i++) {
53                 total_Permutations[*maxSize][i] = array_03[i];
54             }
55             //then maxSize is incremented by 1
56             (*maxSize)++;
57         }
58     }
59     //after all that swap again and check for the sequences that can formed with new order and again will swap and check for seuneces.
60 } else {
61     // generate all permutations by swapping each element with every other element in the array
62     for (int i = front; i <= tail; i++) {
63         swapElements(&array_03[front], &array_03[i]); //swap elements
64         sequence_Generate(array_03, front + 1, tail, total_Permutations, maxSize); //recursively calling function to generate all possible sequences.
65         swapElements(&array_03[front], &array_03[i]); // again swapping
66     }
67 }
68 /*-----end of sequence_Generate function-----*/
69
70
71
72
73
74 /*-----count_arithmetic_sequence function-----*/
75 // Function to count the number of arithmetic sequences in a 2D array of integers
76 int arithmetic_Sequence_count(int **array_03, int maxSize, int array_03_Size) {
77     // variable to count the arithmetic sequence count
78     int sequenceCount = 0;
79     for (int i = 0; i < maxSize; i++) {
80         //taking the difference of the sequence between 2 elements
81         int common_diff = array_03[i][1] - array_03[i][0];
82         int arrayCount = 0;
83
84         //If j is less than array_3 size
85         for (int j = 2; j < array_03_Size; j++) {
86             //taking the difference of the array_03 between 2 elements and checking and adding array count by 1
87             if (array_03[i][j] - array_03[i][j - 1] == common_diff) {
88                 arrayCount++;
89             }
90         }
91         //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 deersing 2 we cheking the rest
92         // of elements*/
93         if (arrayCount == array_03_Size - 2) {
```



```

94     sequenceCount++;
95 }
96 }
97
98 //returning the final value of total arithmetic sequence count
99 return sequenceCount;
100 }
101 /*-----end of count arithmetic sequence function-----*/
102
103
104
105 //start of main body
106 int main() {
107
108     // variable declaration
109     int sizeA, sizeB ;
110
111     // taking the user inputs to initiate the array sizes.
112     printf("\n");
113     printf("-----\n");
114     printf("Enter the size of array A & B      : ");
115     scanf("%d %d",&sizeA , &sizeB);
116     printf("-----\n");
117     //assigning the array sizes that took as user inputs.
118     int array_A[sizeA];
119     int array_B[sizeB];
120     printf("\n");
121
122
123     // taking the user input as array element to array A
124     printf("-----\n");
125     printf("Enter the elements of array A      : ");
126     for(int i = 0; i < sizeA; i++){
127         scanf("%d" , &array_A[i]);
128     }
129     printf("\n");
130
131     // taking the user input as array element to array B
132     printf("Enter the elements of array B      : ");
133     for(int i = 0; i < sizeB; i++){
134         scanf("%d" , &array_B[i]);

```

```

135     }
136     printf("-----\n");
137     printf("\n");
138
139     // Calculating the number of total sequences that can be formed to take the value as the max size of 2D array
140     int maxSequences = 1;
141     for (int i = 1; i <= sizeA + sizeB; i++) {
142         //if condition satisfy adding the multiplying the value of maxsequences by (i)th value
143         maxSequences *= i;
144     }
145
146
147     // allocating the memory dynamically to the 2D array using above obtained values.
148     int **total_Sequences = malloc(maxSequences * sizeof(int *));
149     //The array has maxSequences rows and (sizeA + sizeB) columns. Each row of the array represents a possible
150     //sequence of integers that can be formed by interleaving the elements of two input arrays A and B. The memory for
151     //the array is allocated dynamically using malloc()
152     for (int i = 0; i < maxSequences; i++) {
153         total_Sequences[i] = malloc((sizeA + sizeB) * sizeof(int));
154     }
155
156
157     // this variable is to keep track of number of sequences created.
158     int total_seq = 0;
159
160     //generating sequence from start of first element to end of the array A size
161     for(int a = 0; a < sizeA; a++){
162         for (int b = 0; b < sizeA ; b++){
163             for (int c = 0; c < sizeB ; c++){
164                 sequece_Generate(array_A , 0 , sizeA - 1, total_Sequences, &total_seq);
165
166                 int swappedElements = array_A[b];
167                 array_A[b] = array_B[c];
168                 array_B[c] = swappedElements;
169             }
170         }
171     }
172
173     //generating sequences from the end of the array to upto till 0 th index
174     for(int a = 0; a < sizeA ; a++){
175         for (int b = 0; b < sizeA ; b++){
176             for (int c = 0; c < sizeB ; c++){
177                 sequece_Generate(array_A , 0 , sizeA - 1, total_Sequences, &total_seq);
178
179                 int swappedElements = array_A[sizeA-b];

```

```

180         array_A[sizeA-b] = array_B[sizeB-c];
181         array_B[sizeB-c] = swappedElements;
182     }
183 }
184 }
185
186 //printing the final output (no of unique arithmetic sequence of length of array A)
187 printf("The unique arithmetic sequence count is : %d\n" , arithmetic_Sequence_count(total_Sequences, total_seq , sizeA));
188 printf("-----\n");
189 printf("\n");
190
191 //releasing memory because i used dynamic variables and pointers
192 for (int i = 0; i < maxSequences; i++) {
193     free(total_Sequences[i]);
194 }
195 free(total_Sequences);
196
197 return 0;
198 }
199

```

## 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 sequence 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 sequence 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 sequence 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 variable called swapped
element and assigned the value of element 01 to temp array. then swapped
bothe elements and assigned 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_Duplicate_Sequence_Count function-----*/
-----*/
// Function to check if an array of integers is a duplicate of any previously
generated permutations
bool duplicate_Sequence_Count(int **total_Permutations, int maxSize, int *arr,
int n) {
    for (int i = 0; i < maxSize; i++) {
        //declaring boolean variable to keep track of duplicate sequence count.
        //initially assigning it to true.
        bool dupFound = true;
        /*this for loop has i condition it will check the total_Permutations each
        elements with the given array elements.
        if any single element found in total_Permutations that doesn't match with
        given array element then loop will end and assigned 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 sequence is found in any row, the function returns
false(0).
return false;
}

/*-----end of
Count_Duplicate_Sequence_Count function-----*/

/*-----sequence_Generate
function-----*/

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

    // if the front and tail equal that mean we have sequence 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 (!duplicate_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 sequences that can formed
        with new order and again and again will swap and check for sequences.
    } 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
            sequence_Generate(array_03, front + 1, tail, total_Permutations,
maxSize); //recursivly calling function to generate 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_Sequence_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 seugnce 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 and 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() {

```

```

// variable declaration
int sizeA, sizeB ;

    // taking the user inputs to initiate 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 B
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 multiplying the value of
maxsequences by (i)th value
        maxSequences *= i;
    }

    // allocating the memory dynamically 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 sequece count is      :%d\n" ,
arithmetic_Sequence_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;  
}
```