DSA LAB FAT

Meher Shrishti Nigam 20BRS1193

Q1) Consider the set of integers from 1 up to k for some k>1. Write a program to print all the permutation of length k for the number 1234...k such that the ith digit is always less than the (i+1)th digit for 2<=i<=(k-1).

Input format:

Enter the value of k

Output format:

Permutation of digit up to length k

Code:

```
#include <stdio.h>

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

    for(int i = 1; i <= k; i++)
    {
        printf("%d", i);
        for(int j = 1; j <= k; j++)
        {
            if(j != i)
                printf("%d", j);
        }
        printf("\n");
    }
}
```

Algorithm:

We work out the algorithm, the first element in output can be any number and the rest have to be in ascending sorted order. As number of elements to be printed are k, we will just print all the integers till k in ascending order execpt the first one. Number of permutations will be k.

Step 1: Take input in var k

Step 2: Set outer loop – i from 1 to k (incl)

Step 3: Print i

Step 4: Set inner loop – j from 1 to k(incl)

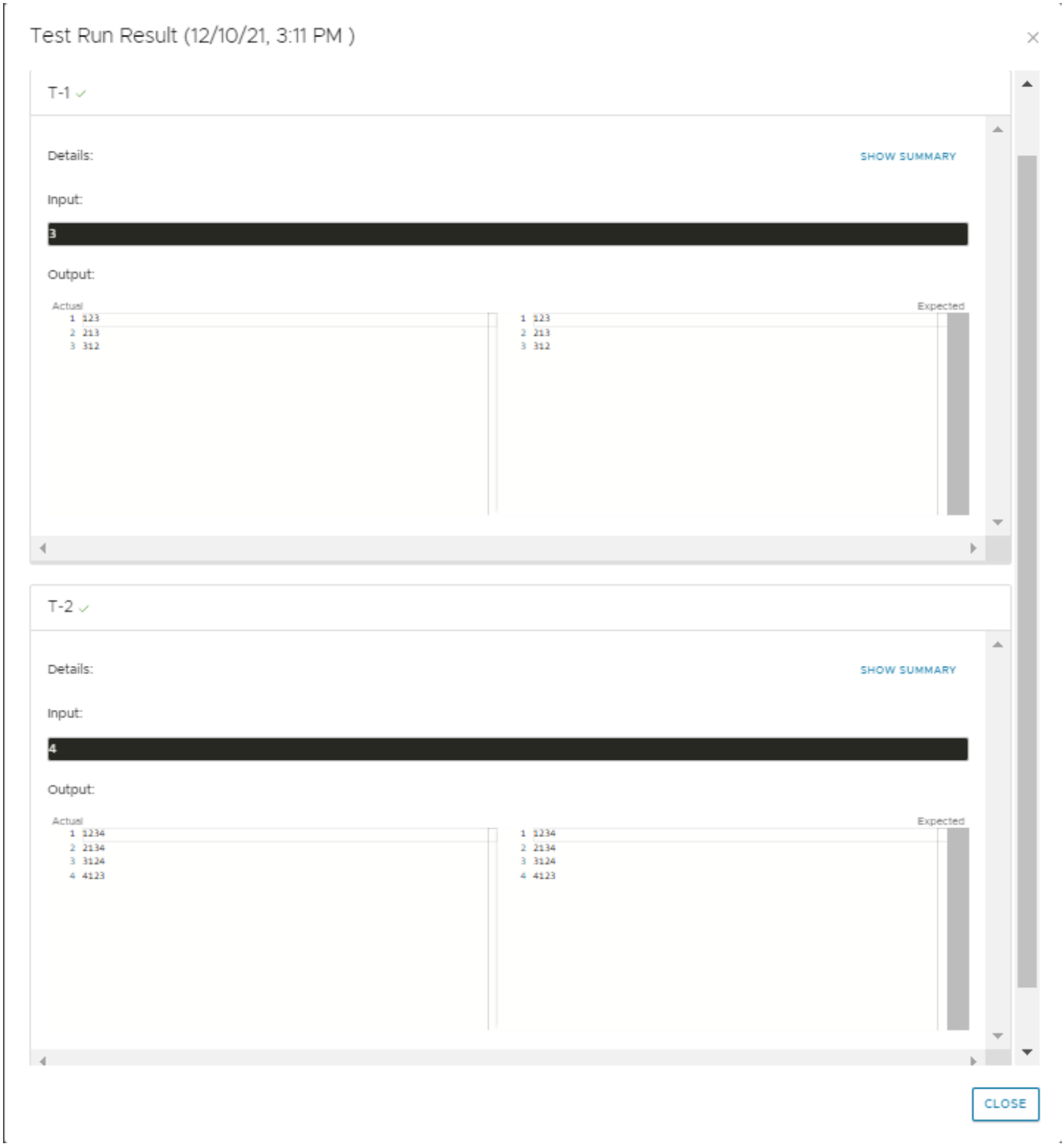
Step 5: if j != i, Print j

Step 6: End inner loop

Step 7: End outer loop

Step 8: End

Output:



Q2) Let L be a linked list with two or more elements. Also, let L={l1,l2,l3,…,lm} where each lj, 1<=j<=m is an element of L occurring in that order in L. Let A[1…n] be an array of n integers such that, the elements of A are a subset of the elements of L. Write a program that will check that the following is true and output the number of occurrences of such pairs: A[i],A[i+1]=ljlj+1 for some i and some j not necessarily i=j with 1<=i<=n.

Input Format:

Total number of elements in linked list: m

Enter elements of linked list

Total number of elements in array: n

Enter elements of array

Output format:

Number of occurrence of pair

Code:

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

typedef struct Node{
    int data;
    struct Node * next;
}Node;

typedef struct SinglyLinkedList
{
    Node * start;
    int len;
}SinglyLinkedList;

Node * createNode(int item)
{
    Node * temp = (Node *)malloc(sizeof(Node));
    temp->data = item;
    temp->next = NULL;
    return temp;
}
```

```
SinglyLinkedList * createSinglyLinkedList()
{
    SinglyLinkedList * sll = (SinglyLinkedList *)malloc(size
of(SinglyLinkedList));
    sll->len = 0;

    /* Creating a start node that will store the location of
 the first node.
     * It stores null at the time of declaration.
                */
    sll->start = (Node *)malloc(sizeof(Node));
    sll->start-
>data = INT_MIN; // Data in the start node is never to be ac
cessed. If INT_MIN is the
                                    // data displayed, an error
has possibly occured
    sll->start->next = NULL;
    return sll;
}

/* Insertion */

// Inserts an element at the end of a linked list
void insertAtEnd(SinglyLinkedList * sll, int item)
{
    Node * newnode = createNode(item);
    Node * ptr = sll->start;
    while(ptr->next != NULL)
        ptr = ptr->next;
    ptr->next = newnode;
    sll->len++;
}

/* Traversal */
// Returns the 1-
base position of the first occurance of data.
Node * search(SinglyLinkedList * sll, int item)
{
    int count = 1;
    Node * ptr = sll->start->next;
    while(ptr->data != item)
```

```
    {
        ptr = ptr->next;
        count++;
        if(ptr->data != item && ptr->next == NULL)
        {
            return NULL;
        }
    }
    return ptr;
}

int main()
{
    SinglyLinkedList * sll = createSinglyLinkedList();
    int n, m;
    scanf("%d", &m);
    for(int i = 0; i < m; i++)
    {
        int x;
        scanf("%d", &x);
        insertAtEnd(sll, x);
    }

    scanf("%d", &n);
    int arr[n];
    for(int i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    int count = 0;
    for(int i = 0; i < n-1; i++)
    {
        Node * ptr = search(sll, arr[i]);
        if(ptr->next->data == arr[i+1])
            count++;
    }
    printf("%d", count);
}
```

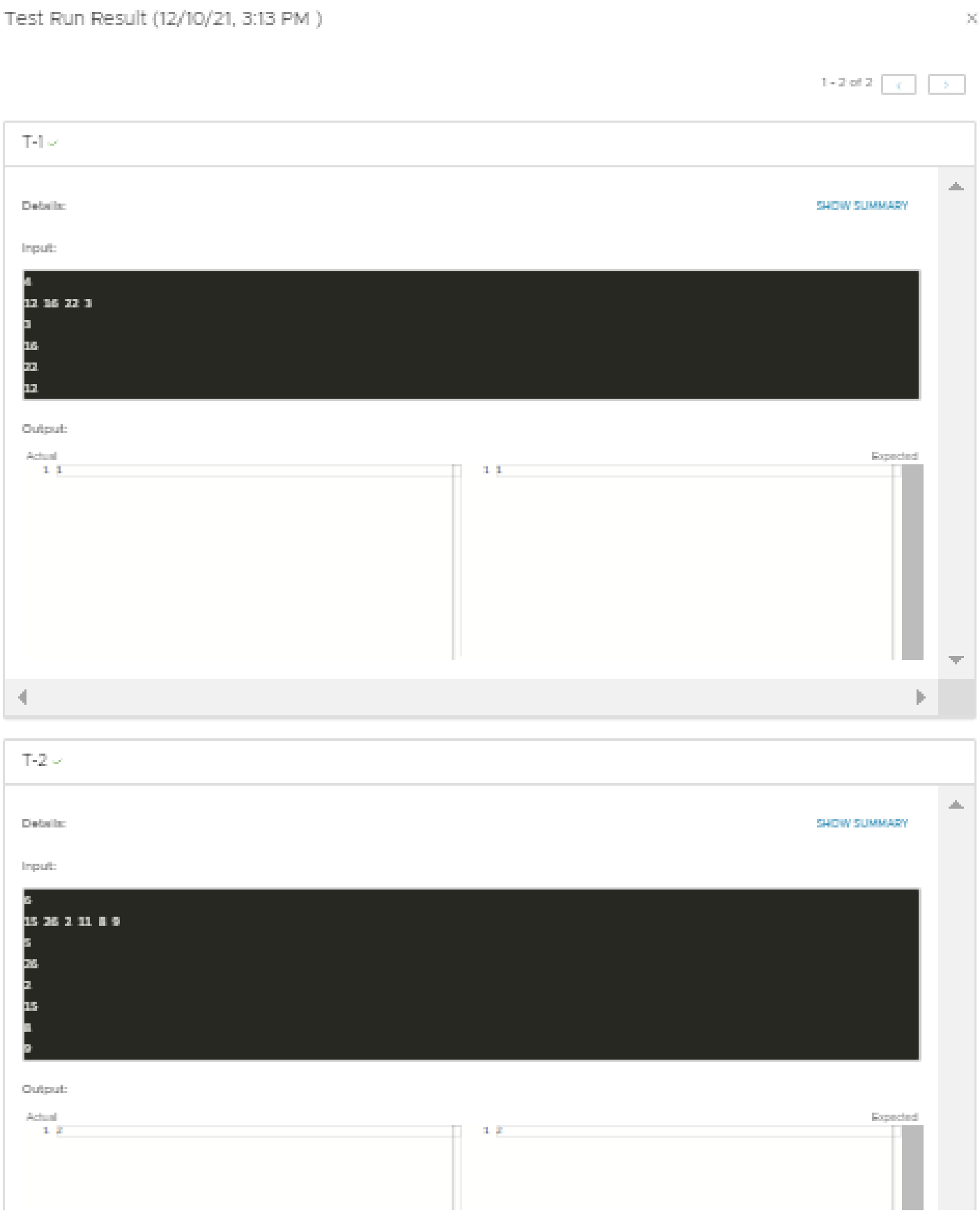
Algorithm:

We first create a singly linked list data structure. We create the Node and SinglyLinkedList structs and createSinglyLinkedList, insertionAtEnd, search functions. The search function returns a Node pointer. We try to match the next element of the array and linked list, if they do match we increment the counter. At the end we display the count.

Step 1: Take input var m
Step 2: Take m elements as input in the singly linked list.
Step 3: Take input var n
Step 4: Take n elements as input in array
Step 5: Set count = 0
Step 6: Start loop: For i from 0 to n-1,
Step 7: Search for arr[i] in singly linked list, store Node in Node * ptr
Step 7: If ptr->next->data == arr[i+1]
Step 8: Increment count
Step 9: End if
Step 10: End loop
Step 11: Display count
Step 12: End

Output in next page!

Output:


Test Run Result (12/10/21, 3:13 PM)
1 - 2 of 2
T-1
Details:
SHOW SUMMARY
Input:
4
12 16 22 3
3
16
22
12
Output:
Actual
1 1
Expected
1 1
T-2
Details:
SHOW SUMMARY
Input:
6
15 26 2 11 8 9
5
26
2
15
8
9
Output:
Actual
1 2
Expected
1 2

Closer Screenshot of tests: