Indian Institute of Technology Ropar

Data Structures and Algorithms

CS506 and CS205

Lab Assignment 2

DEADLINE: 16th August 2025

Read the assignment carefully. There are 3 exercises in the assignment. 3 programs to be submitted.

- Single .c file for each exercise. The program must be compiled using gcc as we will be likely using the linux platform to evaluate your assignments.
- You **MUST** follow the Filename format :
    - EntryNo_DSA25_L02_A.c for part A,
        - For e.g.2025AIM1001_DSA25_L02_A.c for part A exercise
    - EntryNo_DSA25_L02_B.c for part B,
        - For e.g.2025AIM1001_DSA25_L02_B.c for part B exercise
    - EntryNo_DSA25_L02_C.c for part C,
        - For e.g.2025AIM1001_DSA25_L02_C.c for part C exercise
- Input outputs that you tried or used for testing your code can be included in your program file at end within comments (using /* */).
- If you know you code is not working for some cases etc, do mention that clearly in comments section at the beginning of your code (using /* ….. */).
- There shall not be any identifiable information within the program. No reference to your name, roll number etc. shall appear in your code.
- No Plagiarism Please. Please note that we may do plag check now or later even after endsem, and you will be then (**severely**) penalized. You must do your own coding and neither take nor provide codes from/to anyone else.
    - Those who complete their works are kindly requested to not share their codes in the name of friendship or so. You may get severely penalized for the same and by sharing codes, you are not really helping them either.
    - We sincerely request and hope that you will try your level best and not adopt any unfair means so that we need NOT go through the painful experience of penalizing you.
    - By submitting the assignment, you agree that if any plagiarism is found at any point during your studies here, you are willing to accept the severe penalty of even upto the F grade in the course.
- Do not have any unnecessary or redundant input argument.
- Follow strictly the input / output format.

PROGRAMMING LANGUAGE: C

==========EXERCISE A (Using Linked List concepts) ==========

Programming Assignment (Very Basic Assignment):

1. Write a program to perform basic operations as mentioned below on a linked list, where each node consists of an integer data value and a pointer to the next node. The program would ask for user choice (an integer value) and perform the operation as per user choice. Assume "head" is the pointer pointing to the head of the linked list. Initially it is Null as the linked list is empty. You may preferably write all these operations within the main function only. You may write different functions for different operations (It is recommended but it is not mandatory.

Choice	Operation	Details
0	Exit	Exit from the program. A good programmer would free all the space before exiting
1	Insert-in beginning	The user is prompted to input data value and then a new node with that data value is added in the beginning of the linked list.
2	Insert-at end	The user is prompted to input data value and then a new node with that data value is added at the end of the linked list.
3	Delete first node	Delete first node of the linked list. If no such node, print "**Cannot delete as NO nodes in the linked-list**"
4	Delete last node	Delete last node of the linked list. If no such node, print "**Cannot delete as NO nodes in the linked-list**".
5	Delete specific node	The user is prompted to input data value and then there is a search to find the node with that data value in the linked list. If there is any such one node in the linked list, delete that node from the linked list. If there are many such nodes, consider only the first such node. If no such node exists, print "**Cannot delete as no such node in the linked-list**".
6	Find node	The user is prompted to input data value and then there is a search to find the node with that data value in the linked list. If there is any such one node in the linked list, print the position of that node in the linked list and also how many units apart the allocated memory of this node is from that of the current head of the linked list.. If there are many such nodes, consider the first occurrence that is identified. If no such node exists, print "**Cannot find any such node in the linked-list**".

7	Size	Print the number of nodes in the linked list. Print 0 if no nodes in the linked list or say when head==NULL
8	Display	Print all nodes values in the linked list in the manner as mentioned below in the sample input/output below If linked list is empty, Print “Empty”
Any other integer	Wrong Input	Print “**Wrong Input. \nEnter your choice again:** ”

Sample Input / Output (Note that user input values are mentioned in bold here for your convenience):

```
Enter the choice:
1
Enter the data value:
10
Enter the choice:
2
Enter the data value:
20
Enter the choice:
1
Enter the data value:
16
Enter the choice:
7
        Size of the linked list = 3
Enter the choice:
8
        Linked list: 16 => 10 => 20
Enter the choice:
9
        Wrong Input.
Enter your choice again:
3
Enter the choice:
8
        Linked list: 10 => 20
Enter the choice:
2
Enter the data value:
15
Enter the choice:
```

1
Enter the data value:
40
Enter the choice:
4
Enter the choice:
8
Linked list: 40 => 10 => 20
Enter the choice:
6
Enter the data value:
10
Value 10 node is at position 2 in the linked list.
Difference in its allocated memory to that of current head : -6
Enter the choice:
5
Enter the data value:
10
Enter the choice:
6
Enter the data value:
10
Cannot find any such node in the linked-list
Enter the choice:
8
Linked list: 40 => 20
Enter the choice:
7
Size of the linked list = 2
Enter the choice:
0

==========EXERCISE B (Use Linked List concepts) ==========

Consider a new function **#** (somewhat related to factorial !) that is defined for any non-negative integer **n** as follows:

n# = (1^1) * (2^2) * (3^3) * .....* ((n-1) ^ (n-1)) * (n^n)

//Remember that **n!** = 1*2*3*4*…..*(n-1)*n and is different **0# = 1 1#=1 2#=4 2!=2**

Given a non-negative number n and positive number k (at most three digit), Write a program in C that computes and provides the value of n# ,
Z_n i.e. the number of end zeroes in n# value, and
$Q_{n,k}$ i.e. the number of times k pattern appears in n#.

Constraints and Sample input/output -

- The first line of the input file mentions a positive value T that denotes the number of test cases.
- Then follows T lines and each of them mentions two values (single space separated) n_i and k_i

Constraints/Assumptions:

- All these input values are non-negative.
- T<500, n_i <= 200 and k_i <= 999

Input would be read from terminal i.e. Stdin and not from any file.

Input Format: T N_1 k_1 N_2 k_2 ... ... N_T k_T	**Output format (T lines each having 3 values, single space separated):** Z_{N1} $Q_{N1,k1}$ N_1# Z_{N2} $Q_{N2,k2}$ N_2# ... ... Z_{NT} $Q_{NT,kT}$ N_T#
Sample Input: 6 3 1 5 7 9 941 9 09 7 3 9 22	**Sample Output** 0 1 108 5 0 86400000 5 2 215779412229418562091680268288000000 5 1 215779412229418562091680268288000000 5 3 3319766398771200000 5 2 215779412229418562091680268288000000

==========EXERCISE C (Use Stack Concepts) ==========

Given inputs (unsigned integers) N and k , print the number $\mathbf{TSP}_N$ i.e. **T**otal number of **S**tack **P**ermutations possible with these N different numbers (1,2,3,....N), and then the k^{th} stack permutation i.e. k^{th} sequence (in lexicographic ordering) in stack permutations possible using N numbers observed in the order as follows: 1,2,3,…,N. (N would be less than equal to 30)
//T is no. of test cases . Also note that if kth stack permutation is not possible, you may print -1.

Constraints and Sample input/output -

- The first line of the input file mentions a positive value T that denotes the number of test cases.
- Then follows T lines and each of them mentions two values (single space separated) N_i and k_i

Constraints/Assumptions:

- All these (unsigned) input values are non-negative.
- T<500, N_i <= 30 and k_i < 999999

Input would be read from terminal i.e. Stdin and not from any file.

Input Format: T N_1 k_1 N_2 k_2 ... ... N_T k_T	**Output format (T rows each having 1_Ni values, single space separated):** TSP_{N1} <k_1^{th} Stack permutation….N_1 numbers single space separated> TSP_{N2} <k_2^{th} Stack permutation….N_2 numbers single space separated> ... ... TSP_{NT} <k_T^{th} Stack permutation….N_T numbers single space separated>
Sample Input: 4 2 2 3 2 3 5 3 6	**Sample Output** 2 2 1 5 1 3 2 5 3 2 1 5 -1

Note:

A Google form for submitting **assignment code** will be provided soon.

For any clarification, you may contact the TAs in the lab sessions:

Maninder Kaur, Raman Kumar, Amber Gupta, Rizwan Ahmad, Yash Narnaware

ALL THE BEST