Name - Anurag Sarva
Roll No - cs24mtech14003
Subject - ADSA

Code Documentation

1_dp.c (Longest Palindrome Subsequence)

Overview :-
This program in C language will calculate the longest palindrome subsequence in the user input string of alphabetic characters from A to B or a to b.
It will store each of the characters of the input string in a doubly linked list and then it will uses a dynamic programming approach to finding and displaying the length and the content of the longest palindrome in the string provided by the user.

Key Steps and Logic used in the code :-

1. Input Handling and Linked List Creation - This program will construct the doubly linked list (struct input_str) to store each of the characters in the input string.
2. Dynamic Programming Table - A two dimensional array ‘dp’ is used to store the lengths of the palindrome subsequence of the substring, and calculates using a bottom up approach.
3. Filling the DP Table - The DP table will contain the population by comparing each of the characters and going to store the results for subproblems, which helps to build up the solution to finding the longest palindrome subsequence.
4. Result Construction and Output - The longest palindrome subsequence length will be printed, which is followed by reconstructing and printing by the subsequence with the help of the DP table.
5. Memory Cleanup - All dynamically allocated memory like in linked list and DP table, is going to be freed before the program gets terminated.

Function Documentation:-
main() :-
* Purpose - It derives the program by taking an input, and creating a doubly linked list to store the input characters, and then calculating the longest palindrome subsequence, and then printing the result.
* Inputs - users have to give character as an input.
* Outputs - Then it will printing the length and content of the longest palindromic subsequence in a string.

```
(.venv) anurag@anurag-Inspiron-15-5518:~/ACN_Work/ADSA_Work/assign_2$ gcc -o longest_palindrome 1_dp.c
(.venv) anurag@anurag-Inspiron-15-5518:~/ACN_Work/ADSA_Work/assign_2$ ./longest_palindrome
Enter a valid input string (A to Z or a to z), to check the longest palindromic subsequence(enter characters continuously, without any enter): asdfdssa
The length of the longest palindromic subsequence is 7
Longest palindromic subsequence is: asdfdsa
```

2_greedy.c (Optimal Station Placement)

Overview :-
This program helps me to determine the minimum number of stations needed to cover all the towns which are in sequence, and it is located along a line, such that each of the stations can cover all the towns within a maximum distance ‘d’. And then it stores the town locations in a doubly linked list and then uses a greedy algorithm to calculate the optimal number of stations required.

Key Steps and Logic used in this problem :-
1. Town Input and Linked List Creation - My program will going to store each of the town's location in a doubly linked list (struct towns_dll_node), then ensuring the towns are in sorted order.
2. Optimal Station Placement Calculation - Here i am using an greedy approach to solve the problem, this program will iteratively place stations, and ensuring that each of the stations going to cover the maximum number of towns within distance a ‘d’.
3. Result Display - This program will output each of the station's location and also the total count of the station needed.
4. Memory Cleanup - After the task is completed then all the dynamically allocated memory for the linked list is freed up before the program gets terminated.

Function Documentation :-
main() :-
Purpose - The main function takes town locations as input and its last element represent coverage distance as an input, then it calculates the optimal station placement, and then displays the minimum number of stations required.
Inputs - users have to give input as per prompt printed.
Outputs - In output it will print all the locations of each station and then also print the total number of stations required.