Learning With Ram

Video solution link- https://youtu.be/oXejbh62F40?si=j-9My_OV42DKgp8Y

TCS NQT-2025 - 31st march 2025(1st shift coding questions)

Q1)Write a program that reads a string input from the user and removes all consecutive duplicate characters while maintaining the original order of distinct characters. The program should then output the modified string.

Eg:-

Input - 223334566777

Output- 234567

Code:-

```
#include <iostream>
#include <string>
using namespace std;

string removeConsecutiveDuplicates(const string& input) {
    if (input.empty()) return "";
    string result;
    result += input[0]; // Add the first character
    for (size_t i = 1; i < input.length(); ++i) {
        if (input[i] != input[i - 1]) {
            result += input[i];
        }
    }
    return result;
}

int main() {
    string userInput;
    cout << "Enter a string: ";
    cin >> userInput;
    string modifiedString = removeConsecutiveDuplicates(userInput);
```

```
    cout << "Modified string: " << modifiedString << endl;
    return 0;
}
```

Q2)You are given an array of integers and a target sum h. Your task is to determine whether any subset of the given numbers can sum up to h. If a valid subset exists, print "Yes"; otherwise, print "No".

Constraints:

- 1 ≤ t ≤ 100 (Number of elements in the array)
- 1 ≤ timeslot[i] ≤ 1000 (Values in the array)
- 1 ≤ h ≤ 10000 (Target sum)

Example Run:

Enter number of elements in timeslot[]: 4

Enter 4 elements:

3 5 7 2

Enter the target sum (h): 10

Output:

Yes

(Subset {3, 7} sums to 10)

Code:-

Recursive approach:-

```
#include <iostream>
#include <vector>

using namespace std;
```

```
// Function to check if there exists a subset with the given target sum
bool subsetSum(const vector<int>& nums, int index, int target) {
    // Base case: If target sum is achieved, return true
    if (target == 0) return true;
    // Base case: If no elements are left to check, return false
    if (index < 0) return false;

    // If current element is less than or equal to target, try including it
    if (nums[index] <= target && subsetSum(nums, index - 1, target - nums[index]))
        return true;
    // Exclude the current element and check for the remaining elements
    return subsetSum(nums, index - 1, target);
}

int main() {
    int n, target;

    // Input: Number of elements in the array
    cout << "Enter number of elements: ";
    cin >> n;
    vector<int> nums(n);

    // Input: Elements of the array
    cout << "Enter elements: ";
    for (int &num : nums)
        cin >> num;
    // Input: Target sum to check
    cout << "Enter target sum: ";
    cin >> target;
    // Output: Print whether a subset with the target sum exists or not
    cout << (subsetSum(nums, n - 1, target) ? "Yes" : "No") << endl;
```

```
 /*  if(subsetSum(nums,n-1,target))
   {
      cout<<"yes";
   }
   else
   {
      cout<<"NO"
   } */
   return 0;
}
```

Using Dynamic programming

```
/* Approach
The program takes t as input, representing the number of elements in the array.
It reads t integers into a vector timeslot[].
It reads the target sum h.
The solve() function uses Dynamic Programming to determine if a subset sum exists.
If a subset exists, it prints "Yes"; otherwise, it prints "No".
*/

#include <iostream>
#include <vector>
using namespace std;

// Function to check if there exists a subset with sum equal to h
bool solve(vector<int>& a, int h) {
   vector<bool> dp(h + 1, false);
   dp[0] = true; // Base case: sum of 0 is always possible

   for (int num : a) { // Iterate over each number in the array
      for (int j = h; j >= num; j--) { // Traverse backward to prevent duplicate use
```

```
            if (dp[j - num]) {
                dp[j] = true;
            }
        }
    }
    return dp[h]; // Return if h can be achieved
}

int main() {
    int t, h;
    cout << "Enter number of elements in timeslot[]: ";
    cin >> t;

    vector<int> timeslot(t);
    cout << "Enter " << t << " elements:" << endl;
    for (int i = 0; i < t; i++) {
        cin >> timeslot[i];
    }
    cout << "Enter the target sum (h): ";
    cin >> h;

    if (solve(timeslot, h))
        cout << "Yes" << endl;
    else
        cout << "No" << endl;

    return 0;
}
```