[@KLWKS_BOT] THANOS

Date of the Session:__/__/_____ **Time of the Session:_____to_____**

EX – 7 Working with Dynamic Programming

Prerequisites:

- Basics of Data Structures and C Programming.
- Basic knowledge about Arrays and Binary Search Tree.

Pre-Lab:

1) A student named Satish is eagerly waiting to attend tomorrow's class. As he searched the concepts of tomorrow's lecture in the course handout and started solving the problems, in the middle he was stroked in the concept of strings because he was poor in this concept so help him so solve the problem, given two strings str1 and str2 and below operations that can performed on str1. Find minimum number of edits (operations) required to convert 'str1' into 'str2'.
   1. Insert
   2. Remove
   3. Replace

   Input
   str1 = "cat", str2 = "cut"
   Output
   1
   We can convert str1 into str2 by replacing 'a' with 'u'.
   Input
   str1 = "sunday", str2 = "saturday"
   Output
   3

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

int min(int x, int y, int z) {
   if (x <= y && x <= z)
      return x;
   if (y <= x && y <= z)
      return y;
   return z;
}

int minEditDistance(char *str1, char *str2) {
   int m = strlen(str1);
   int n = strlen(str2);

   int dp[m + 1][n + 1];

   for (int i = 0; i <= m; i++)
      dp[i][0] = i;
```

```
    for (int j = 0; j <= n; j++)
        dp[0][j] = j;

    for (int i = 1; i <= m; i++) {
        for (int j = 1; j <= n; j++) {
            if (str1[i - 1] == str2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1];
            } else {
                dp[i][j] = 1 + min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1]);
            }
        }
    }

    return dp[m][n];
}

int main() {
    char str1[100], str2[100];
    printf("Enter the first string: ");
    scanf("%s", str1);
    printf("Enter the second string: ");
    scanf("%s", str2);
    printf("The minimum edit distance is: %d\n", minEditDistance(str1, str2));
    return 0;
}
```

[@KLWKS_BOT] THANOS

2) Given N numbers n1,n2,…nN and Q queries q1,q2,…qQ. Your task is to print Q (Q< j <numbers) f1,f2,…fQ, corresponding to query qj1 max(n1=fj ,n2 ,...nq) using dynamic programming.

Input

5 3
5 4 8 6 9
2 3 5

Output

5 8 9

```
#include <stdio.h>

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

   int nums[N], queries[Q], max_up_to[N];

   for (int i = 0; i < N; i++) {
      scanf("%d", &nums[i]);
   }

   for (int i = 0; i < Q; i++) {
      scanf("%d", &queries[i]);
   }

   max_up_to[0] = nums[0];
   for (int i = 1; i < N; i++) {
      max_up_to[i] = (nums[i] > max_up_to[i - 1]) ? nums[i] : max_up_to[i - 1];
   }

   for (int i = 0; i < Q; i++) {
      int query = queries[i] - 1;
      printf("%d ", max_up_to[query]);
   }
   printf("\n");

   return 0;
}
```

[@KLWKS_BOT] THANOS

In-Lab:

1) Bhanu is a student at KL University who likes playing with strings, after reading a question from their lab workbook for the DAA Course she found what is meant by a subsequence.

(A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without changing the order of the remaining elements)

So, she created 2 strings out of which one she considered as a master string and the other one as a slave string. She challenged her friend Teju to find out whether the slave string is a subsequence of the master string or not, As Teju is undergoing her CRT classes she decided to code the logic for this question. Help her in building the logic and write a code using Dynamic programming concept.

Source code:

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>

bool is_subsequence(char *master, char *slave) {
   int m = strlen(master), n = strlen(slave);
   bool dp[m + 1][n + 1];

   for (int i = 0; i <= m; i++)
      dp[i][0] = true;
   for (int j = 1; j <= n; j++)
      dp[0][j] = false;

   for (int i = 1; i <= m; i++) {
      for (int j = 1; j <= n; j++) {
         if (master[i - 1] == slave[j - 1])
            dp[i][j] = dp[i - 1][j - 1];
         else
            dp[i][j] = dp[i - 1][j];
      }
   }

   return dp[m][n];
}

int main() {
   char master[] = "abcdef";
   char slave[] = "ace";

   if (is_subsequence(master, slave))
```

```
        printf("True\n");
    else
        printf("False\n");

    return 0;
}
```

2) Raju has very less Marks in Dynamic programming test conducted by his teacher. So, the teacher had given a problem about optimal Binary Search Tree which should be solved using Dynamic Programming by the next class. Since, he is very weak in Dynamic programming you must help him to solve the problem. The problem can be described as follows:
An unsorted data of keys and their frequencies are given to you and some queries where each query contains two integers which describe the range of the indices (index range) for which you must print the root of that Optimal Binary Search Tree.

Input

4
12 10 20 21
8 34 50 1
2
0 3
0 1

Output

Cost of Optimal BST is 144
20
1

Source code:

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

int sum(int freq[], int i, int j) {
   int total = 0;
   for (int k = i; k <= j; k++) {
      total += freq[k];
   }
   return total;
}

void optimal_bst(int keys[], int freq[], int n) {
   int dp[n][n];
   int root[n][n];

   for (int i = 0; i < n; i++) {
      dp[i][i] = freq[i];
      root[i][i] = i;
   }

   for (int len = 2; len <= n; len++) {
      for (int i = 0; i <= n - len; i++) {
         int j = i + len - 1;
         dp[i][j] = INT_MAX;
```

```
        int total_freq = sum(freq, i, j);

        for (int r = i; r <= j; r++) {
            int cost = (r > i ? dp[i][r - 1] : 0) + (r < j ? dp[r + 1][j] : 0) + total_freq;

            if (cost < dp[i][j]) {
                dp[i][j] = cost;
                root[i][j] = r;
            }
        }
    }
  }

  printf("Cost of Optimal BST is %d %d\n", dp[0][n - 1], keys[root[0][n - 1]]);
  printf("%d\n", root[0][n - 1]);
}

int main() {
  int keys[] = {12, 10, 20, 21};
  int freq[] = {8, 34, 50, 1};
  int n = sizeof(keys) / sizeof(keys[0]);

  optimal_bst(keys, freq, n);

  return 0;
}
```

[@KLWKS_BOT] THANOS

Post-Lab:

1) Bhanu and Teju are playing dice game where there are N dice with M faces and the dice are numbered from 1 to M. A person wins the game if the sum of the faces of dice adds up to a value X. you are playing on Bhanu's team, and It is Teju's turn now.
You are supposed to find number of ways your opponent can win the game where N, M and X are provided as input. Use Dynamic programming to solve the problem.
Using DP (Dynamic programming) to find the number of ways to get sum X.
Input
M = 2
N = 2
X = 3
Output
2

Source code:

```
#include <stdio.h>
#include <string.h>

int countWays(int M, int N, int X) {
   int dp[N + 1][X + 1];
   memset(dp, 0, sizeof(dp));

   dp[0][0] = 1;

   for (int i = 1; i <= N; i++) {
      for (int j = 1; j <= X; j++) {
         dp[i][j] = 0;
         for (int k = 1; k <= M; k++) {
            if (j - k >= 0) {
               dp[i][j] += dp[i - 1][j - k];
            }
         }
      }
   }

   return dp[N][X];
}
int main() {
   int M = 2;
   int N = 2;
   int X = 3;

   printf("%d\n", countWays(M, N, X));
   return 0;
}
```

[@KLWKS_BOT] THANOS

Comments of the Evaluators (if Any)	Evaluator's Observation Marks Secured:_____out of [50]. Signature of the Evaluator Date of Evaluation: