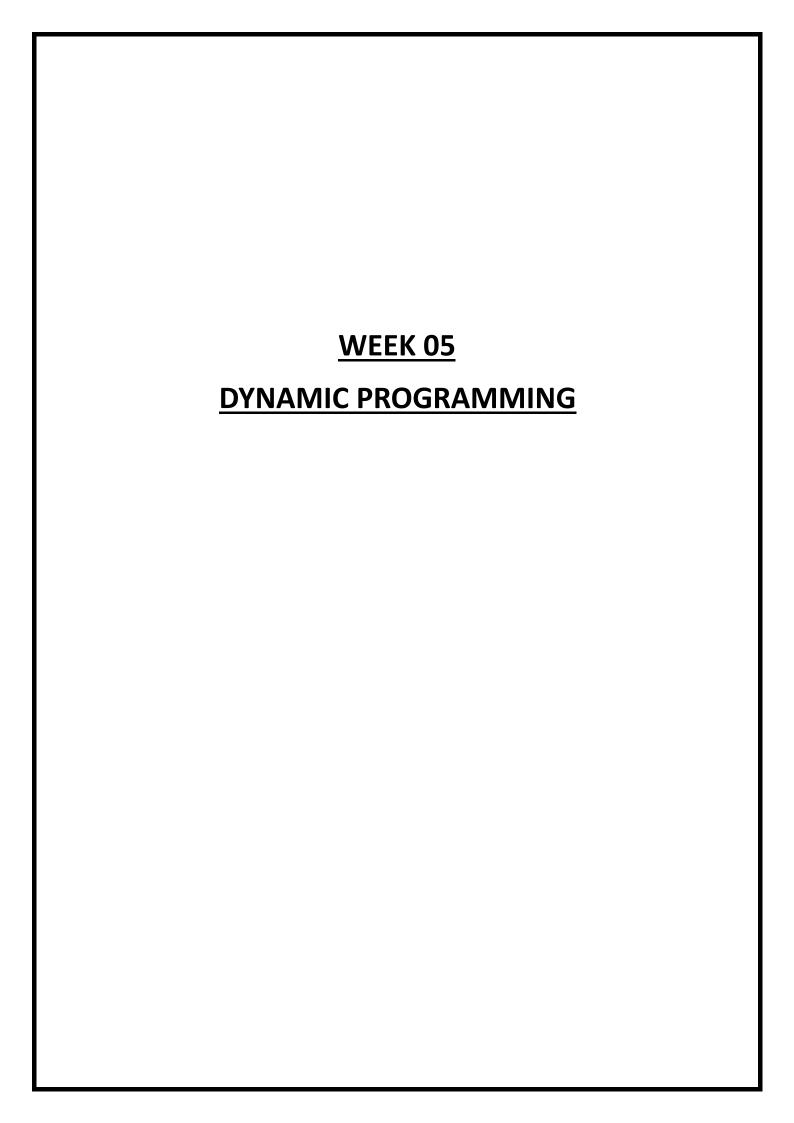
RAJALAKSHMI ENGINEERING COLLEGE RAJALAKSHMI NAGAR, THANDALAM – 602 105



CS23331 DESIGN AND ANALYSIS OF ALGORITHM LAB

Laboratory Observation Notebook

Name: MAHESH KUMAR J R
Year / Branch / Section : 2 nd .Year/ AIML./.B
Register No. :
Semester : 3 rd Semester
Academic Year : 2024-2025



1) Playing with Numbers:

Ram and Sita are playing with numbers by giving puzzles to each other. Now it was Ram term, so he gave Sita a positive integer 'n' and two numbers 1 and 3. He asked her to find the possible ways by which the number n can be represented using 1 and 3. Write any efficient algorithm to find the possible ways.

Example 1:

```
Input: 6
```

Output:6

Explanation: There are 6 ways to 6 represent number with 1 and 3

1+1+1+1+1+1

3+3

1+1+1+3

1+1+3+1

1+3+1+1

3+1+1+1

Input Format

First Line contains the number n

Output Format

Print: The number of possible ways 'n' can be represented using 1 and 3

Sample Input

6

Sample Output

CODE:

```
#include <stdio.h> #include
<stdlib.h> long long
countWays(int n) {    if (n <</pre>
0) return 0; if (n == 0)
return 1;
  long long *dp = (long long*)calloc(n + 1, sizeof(long long));
dp[0] = 1; dp[1] = 1; dp[2] = 1;
  for (int i = 3; i \le n; i++) {
dp[i] = dp[i-1] + dp[i-3];
  }
  long long result = dp[n];
free(dp); return result;
}
int main() {
int n;
  scanf("%d", &n); long long
ways = countWays(n);
printf("%lld\n", ways); return 0;
```

	Input	Expected	Got	
~	6	6	6	~
~	25	8641	8641	~
_	100	24382819596721629	24382819596721629	~

Passed all tests! ✓

Correct

Marks for this submission: 10.00/10.00.

2) Playing with Chessboard:

Ram is given with an n*n chessboard with each cell with a monetary value. Ram stands at the (0,0), that the position of the top left white rook. He is been given a task to reach the bottom right black rook position (n-1, n-1) constrained that he needs to reach the position by traveling the maximum monetary path under the condition that he can only travel one step right or one step down the board. Help ram to achieve it by providing an efficient DP algorithm.

Example:

Input

3

124

234

871

Output:

19

Explanation:

Totally there will be 6 paths among that the optimal is Optimal path value:1+2+8+7+1=19

Input Format

First Line contains the integer n

The next n lines contain the n*n chessboard values

Output Format

Print Maximum monetary value of the path

CODE:

```
#include <stdio.h>
#include <stdlib.h> int
max(int a, int b) {
return (a > b) ? a : b;
}
int findMaxPath(int n, int **board) {
**dp = (int **)malloc(n * sizeof(int *));
for (int i = 0; i < n; i++) { dp[i] = (int)
*)malloc(n * sizeof(int));
  dp[0][0] = board[0][0]; for (int j =
1; j < n; j++) { dp[0][j] = dp[0][j-1]
+ board[0][j];
  }
  for (int i = 1; i < n; i++) {
dp[i][0] = dp[i-1][0] + board[i][0];
  }
  for (int i = 1; i < n; i++) {
for (int j = 1; j < n; j++) {
       dp[i][j] = max(dp[i-1][j], dp[i][j-1]) + board[i][j];
     }
  }
```

```
int result = dp[n-1][n-1];
for (int i = 0; i < n; i++) {
free(dp[i]);
  }
  free(dp);
return result;
}
int main() { int n;
scanf("%d", &n);
  int **board = (int **)malloc(n * sizeof(int *));
for (int i = 0; i < n; i++) {
    board[i] = (int *)malloc(n * sizeof(int));
for (int j = 0; j < n; j++) { scanf("%d",
&board[i][j]);
  }
  int maxPath = findMaxPath(n, board);
printf("%d\n", maxPath);
```

```
// Free the board array
for (int i = 0; i < n; i++) {
free(board[i]);
  }
  free(board);
return 0;
}</pre>
```



3) Given two strings find the length of the common longest subsequence(need not be contiguous) between the two.

Example:

s1: ggtabe

s2: tgatasb

s1 aggtab

s2 g x t x a y b

The length is 4

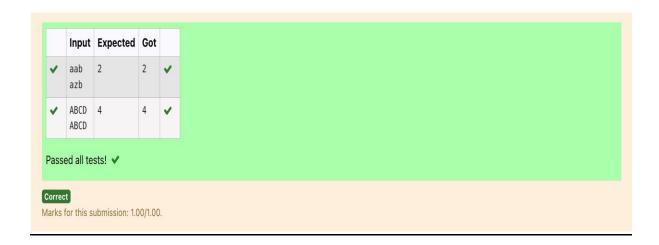
Solving it using Dynamic Programming For example:

Input	Result
aab azb	2

CODE:

#include <stdio.h>
#include <string.h> int
lcs(char *s1, char *s2) {
int m = strlen(s1);

```
int n = strlen(s2);
                      int
dp[m+1][n+1]; for (int i =
0; i \le m; i++) \{ for (int j) \}
= 0; j <= n; j++) {
                         if (i
== 0 | j == 0  {
         dp[i][j] = 0; // LCS of any string with an empty string is 0
       }
       else if (s1[i-1] == s2[j-1]) {
         dp[i][j] = dp[i-1][j-1] + 1; // Characters match
       }
else {
         dp[i][j] = (dp[i-1][j] > dp[i][j-1]) ? dp[i-1][j] : dp[i][j-1]
       }
  }
    return dp[m][n];
int main() { char
s1[100], s2[100];
scanf("%s %s", s1, s2);
int result = lcs(s1, s2);
printf("%d\n", result);
return 0;
}
```



4) Problem statement:

Find the length of the Longest Non-decreasing Subsequence in a given Sequence.

Eg:

```
Input:9
Sequence:[-1,3,4,5,2,2,2,2,3] the
```

subsequence is [-1,2,2,2,2,3]

Output:6

CODE:

```
#include<stdio.h> int
main()
{    int
n,i,j;
    scanf("%d",&n);
int a[n];
for(i=0;i<n;i++)
scanf("%d",&a[i]);
int b[n];
for(i=0;i<n;i++)
b[i]=1;    int max=1;
for (i = 1; i < n; i++) {</pre>
```

```
for (j = 0; j < i; j++) {
    if (a[j] <= a[i]) {
        b[i] = b[i] > (b[j] + 1) ? b[i] : (b[j] + 1);
    }
    if(b[i] > max)

max = b[i];
    }
    printf("%d", max);
}
```

