(1)Question 1

Not complete

Marked out of 10.00

Flag question

Question text

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

6

Answer:(penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

ANSWER:

```c
#include <stdio.h>

long long countWays(int n) {
    if (n == 0) return 1;
    if (n == 1) return 1;
    if (n == 2) return 1;
    if (n == 3) return 2;

    long long dp[n + 1];
    dp[0] = 1;
    dp[1] = 1;
    dp[2] = 1;
    dp[3] = 2;

    for (int i = 4; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 3];
    }

    return dp[n];
}

int main() {
    int n;
    scanf("%d", &n);
    printf("%lld\n", countWays(n));
```

```
    return 0;
}
```

CODE:

Answer: (penalty regime: 0 %)

Time left 0:40:

```c
#include <stdio.h>

long long countWays(int n) {
    if (n == 0) return 1;
    if (n == 1) return 1;
    if (n == 2) return 1;
    if (n == 3) return 2;

    long long dp[n + 1];
    dp[0] = 1;
    dp[1] = 1;
    dp[2] = 1;
    dp[3] = 2;

    for (int i = 4; i <= n; i++) {
        dp[i] = dp[i - 1] + dp[i - 3];
    }

    return dp[n];
}

int main() {
    int n;
    scanf("%d", &n);
    printf("%lld\n", countWays(n));
    return 0;
}
```

	Input	Expected	Got	
✔	6	6	6	✔
✔	25	8641	8641	✔
✔	100	24382819596721629	24382819596721629	✔

(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
1 2 4
2 3 4
8 7 1
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

Answer:

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

long long maxll(long long a, long long b) {
    return (a > b) ? a : b;
}

int main() {
    int n;
    if (scanf("%d", &n) != 1) return 0;
    if (n <= 0) {
        printf("0\n");
        return 0;
    }

    long long *board = malloc((size_t)n * n * sizeof(long long));
    if (!board) return 1;

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            scanf("%lld", &board[i * n + j]);
        }
    }
```

```
long long *dp = malloc((size_t)n * n * sizeof(long long));
if (!dp) {
    free(board);
    return 1;
}

dp[0] = board[0];

for (int j = 1; j < n; ++j) {
    dp[j] = dp[j - 1] + board[j];
}

for (int i = 1; i < n; ++i) {
    dp[i * n + 0] = dp[(i - 1) * n + 0] + board[i * n + 0];
    for (int j = 1; j < n; ++j) {
        long long fromTop = dp[(i - 1) * n + j];
        long long fromLeft = dp[i * n + (j - 1)];
        dp[i * n + j] = board[i * n + j] + maxll(fromTop, fromLeft);
    }
}

printf("%lld\n", dp[(n - 1) * n + (n - 1)]);

free(board);
free(dp);
```

return 0;

}

CODE:

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

long long maxll(long long a, long long b) {
    return (a > b) ? a : b;
}

int main() {
    int n;
    if (scanf("%d", &n) != 1) return 0;
    if (n <= 0) {
        printf("0\n");
        return 0;
    }

    long long *board = malloc((size_t)n * n * sizeof(long long));
    if (!board) return 1;

    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            scanf("%lld", &board[i * n + j]);
        }
    }

    long long *dp = malloc((size_t)n * n * sizeof(long long));
    if (!dp) {
        free(board);
        return 1;
    }

    dp[0] = board[0];

    for (int j = 1; j < n; ++j) {
        dp[j] = dp[j - 1] + board[j];
    }

    for (int i = 1; i < n; ++i) {
        dp[i * n + 0] = dp[(i - 1) * n + 0] + board[i * n + 0];
        for (int j = 1; j < n; ++j) {
            long long fromTop = dp[(i - 1) * n + j];
            long long fromLeft = dp[i * n + (j - 1)];
            dp[i * n + j] = board[i * n + j] + maxll(fromTop, fromLeft);
        }
    }

    printf("%lld\n", dp[(n - 1) * n + (n - 1)]);

    free(board);
    free(dp);
    return 0;
}
```

	Input	Expected	Got	
✔	3 1 2 4 2 3 4 8 7 1	19	19	✔
✔	3 1 3 1 1 5 1 4 2 1	12	12	✔
✔	4 1 1 3 4 1 5 7 8 2 3 4 6 1 6 9 0	28	28	✔

Passed all tests! ✔

(3) **3-DP-Longest Common Subsequence**

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

Example:

s1: ggtabe

s2: tgatasb

s1		a	g	**g**	**t**	**a**	**b**	
s2		**g**	x	**t**	**x**	**a**	**y**	**b**

The length is 4

Solveing it using Dynamic Programming

For example:

Input	Result
aab azb	2

Answer

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

int main() {
    char s1[10005], s2[10005];
    if (scanf("%10004s", s1) != 1) return 0;
    if (scanf("%10004s", s2) != 1) return 0;

    int m = (int)strlen(s1);
    int n = (int)strlen(s2);

    int *dp = malloc((m + 1) * (n + 1) * sizeof(int));
    if (!dp) return 1;

    for (int i = 0; i <= m; ++i) {
        for (int j = 0; j <= n; ++j) {
            if (i == 0 || j == 0) dp[i * (n + 1) + j] = 0;
```

```
            else if (s1[i - 1] == s2[j - 1])
                dp[i * (n + 1) + j] = dp[(i - 1) * (n + 1) + (j - 1)] + 1;
            else {
                int a = dp[(i - 1) * (n + 1) + j];
                int b = dp[i * (n + 1) + (j - 1)];
                dp[i * (n + 1) + j] = (a > b) ? a : b;
            }
        }
    }

    printf("%d\n", dp[m * (n + 1) + n]);
    free(dp);
    return 0;
}
```

CODE:

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

int main() {
    char s1[10005], s2[10005];
    if (scanf("%10004s", s1) != 1) return 0;
    if (scanf("%10004s", s2) != 1) return 0;

    int m = (int)strlen(s1);
    int n = (int)strlen(s2);

    int *dp = malloc((m + 1) * (n + 1) * sizeof(int));
    if (!dp) return 1;

    for (int i = 0; i <= m; ++i) {
        for (int j = 0; j <= n; ++j) {
            if (i == 0 || j == 0) dp[i * (n + 1) + j] = 0;
            else if (s1[i - 1] == s2[j - 1])
                dp[i * (n + 1) + j] = dp[(i - 1) * (n + 1) + (j - 1)] + 1;
            else {
                int a = dp[(i - 1) * (n + 1) + j];
                int b = dp[i * (n + 1) + (j - 1)];
                dp[i * (n + 1) + j] = (a > b) ? a : b;
            }
        }
    }

    printf("%d\n", dp[m * (n + 1) + n]);
    free(dp);
    return 0;
}
```

	Input	Expected	Got	
✔	aab azb	2	2	✔
✔	ABCD ABCD	4	4	✔

Passed all tests! ✔

(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

Answer

```
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    int dp[n];
    int max = 1;

    for (int i = 0; i < n; i++) {
        dp[i] = 1;
        for (int j = 0; j < i; j++) {
            if (arr[j] <= arr[i] && dp[j] + 1 > dp[i])
                dp[i] = dp[j] + 1;
```

```
        }
        if (dp[i] > max)
            max = dp[i];
    }

    printf("%d\n", max);
    return 0;
}
```

CODE:

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    int dp[n];
    int max = 1;

    for (int i = 0; i < n; i++) {
        dp[i] = 1;
        for (int j = 0; j < i; j++) {
            if (arr[j] <= arr[i] && dp[j] + 1 > dp[i])
                dp[i] = dp[j] + 1;
        }
        if (dp[i] > max)
            max = dp[i];
    }

    printf("%d\n", max);
    return 0;
}

```

	Input	Expected	Got	
✔	9 -1 3 4 5 2 2 2 2 3	6	6	✔
✔	7 1 2 2 4 5 7 6	6	6	✔

Passed all tests! ✔