CHANAKYA UNIVERSITY

DSA assignment – 2

Name: - Mohan. M

Reg no: - 24UG00523

Section: - D EE(VLSI)

Question: -

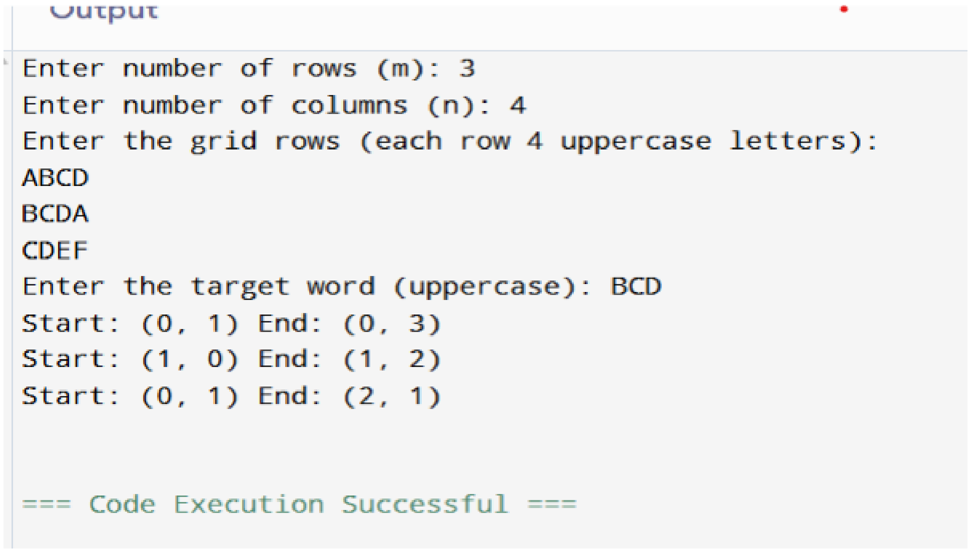
You are required to model a 2D grid of characters as a graph, where each cell acts as a node, and edges exist between horizontally and vertically adjacent cells (no diagonal connections). Given a target word, your program should search for occurrences of the word in the grid horizontally (left to right) or vertically (top to bottom). For each occurrence of the word found, print the start node and end node coordinates. If there are no occurrences, print Word not found.

CODE: -

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
   int m, n;
   printf("Enter number of rows (m): ");
   if (scanf("%d", &m) != 1) {
      fprintf(stderr, "Invalid input for m\n");
      return 1;
   }
   printf("Enter number of columns (n): ");
   if (scanf("%d", &n) != 1) {
      fprintf(stderr, "Invalid input for n\n");
      return 1;
   }
   char **grid = malloc(m * sizeof(char *));
   for (int i = 0; i < m; i++) {
      grid[i] = malloc((n + 1) * sizeof(char));
   }
   printf("Enter the grid rows (each row %d uppercase letters):\n", n);
   for (int i = 0; i < m; i++) {
      scanf("%s", grid[i]);
      if ((int)strlen(grid[i]) != n) {
         fprintf(stderr, "Row %d length is not %d\n", i, n);
         return 1;
      }
   }
   char word[101];
```

```
printf("Enter the target word (uppercase): ");
scanf("%s", word);
int L = strlen(word);
int found = 0;
for (int i = 0; i < m; i++) {
    for (int j = 0; j + L - 1 < n; j++) {
        int k;
        for (k = 0; k < L; k++) {
            if (grid[i][j + k] != word[k]) break;
        }
        if (k == L) {
            printf("Start: (%d, %d) End: (%d, %d)\n", i, j, i, j + L - 1);
            found++;
        }
    }
}
for (int i = 0; i + L - 1 < m; i++) {
    for (int j = 0; j < n; j++) {
        int k;
        for (k = 0; k < L; k++) {
            if (grid[i + k][j] != word[k]) break;
        }
        if (k == L) {
            printf("Start: (%d, %d) End: (%d, %d)\n", i, j, i + L - 1, j);
            found++;
        }
    }
}
if (!found) {
        printf("Word not found\n");
    }
    for (int i = 0; i < m; i++) {
        free(grid[i]);
    }
    free(grid);
    return 0;
}
```

Output:



```
Enter number of rows (m): 3
Enter number of columns (n): 4
Enter the grid rows (each row 4 uppercase letters):
ABCD
BCDA
CDEF
Enter the target word (uppercase): BCD
Start: (0, 1) End: (0, 3)
Start: (1, 0) End: (1, 2)
Start: (0, 1) End: (2, 1)


=== Code Execution Successful ===
```