ASSIGNMENT:02

NEETHUSHREE.C

24UG00527

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): 7
Enter number of columns (n): 7
Enter the grid rows (each row 7 uppercase letters):
NEETHUN
EETHUNE
ETHUNEE
THUNEET
HUNEETH
UNEETHU
NEETHUN
Enter the target word (uppercase): NEETHU
Start: (0, 0) End: (0, 5)
Start: (5, 1) End: (5, 6)
Start: (6, 0) End: (6, 5)
Start: (0, 0) End: (5, 0)
Start: (0, 6) End: (5, 6)
Start: (1, 5) End: (6, 5)


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