```
/*
Q4: Develop a project to implement knights' travails where a knight can keep in any square in the first row
of the chess board and move this knight to the last row with a minimum number of moves.
Use the proper data structures to store the moves and search for the valid moves in an efficient manner.
*/
#include <stdio.h>
#include <stdlib.h>
#define n 8
int  visited[n][n], board[n][n], endCol; // 8 x 8 is the dimension of chess board
// endCol variable will store the column of last row on which knight reached

// queue to store adjacent & next move's x & y co-ordinates
struct queue
{
   int x, y, pos;
```

```
    struct queue *next;
} *front = NULL, *rear = NULL;


void enQueue(int xval, int yval, int p)
{
    struct queue *node = (struct queue *)malloc(sizeof(struct
queue));
    node->x = xval;
    node->y = yval;
    node->pos = p;
    node->next = NULL;
    if (!front)
    {
        front = rear = node;
    }
    else
    {
        rear->next = node;
        rear = rear->next;
```

```
    }
}

struct queue *deQueue()
{
    struct queue *temp = front;
    if (!front)
        return NULL;
    else
    {
        if (rear == front)
            front = rear = NULL;
        else
            front = front->next;

        temp->next = NULL;
        return temp;
    }
```

```
}

// to check if the given co-ordenates (x, y) is inside 8 x 8 chess board or not
int isInside(int x, int y)
{
   if (x < n && x >= 0 && y < n && y >= 0)
      return 1;
   else
      return 0;
}

// to find the next possible move of knight and enqueue it in given queue
void adjacent(int x, int y, int m)
{
   int dx[] = {-2, -1, 1, 2,  2,  1, -1, -2},
      dy[] = { 1,  2, 2, 1, -1, -2, -2, -1}, a,b;
   for (int i = 0; i < 8; i++)
```

```
    {
        a = x + dx[i]; b = y + dy[i];//the current coordinates of
knight
        if (isInside(a, b) && visited[a][b] == -1 && board[a][b]==-1)
        {
            enQueue(a, b, m + 1);
            board[a][b] = m + 1;
        }
    }
}

// (0, y1) --> (7, y2) in min steps
int steps(int begCol)
{
    struct queue *temp;
    int x = 0, y = begCol; // x, y co-ordinate
    // to mark every block of chess board unvisited
    visited[x][y] = board[x][y] = 0;
    adjacent(x, y, 0);
```

```
    while (front)
    {
        temp = deQueue();
        x = temp->x;
        y = temp->y;
        visited[x][y] = 0;
        if (x == n - 1){
            endCol = y;
            return temp->pos;
        }
        adjacent(x, y, temp->pos);
    }
    return -1;
}

//to show the path taken by knight on board
void display()
{
```

```
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if(visited[i][j] != -1){
                printf("  %d", visited[i][j]);
            }
            else
            {
                printf(" %d", visited[i][j]);
            }
        }
        printf("\n");
    }
}
//it will display all possibilities on chess board for each move of knight
// void displayB(){
//     for (int i = 0; i < n; i++)
```

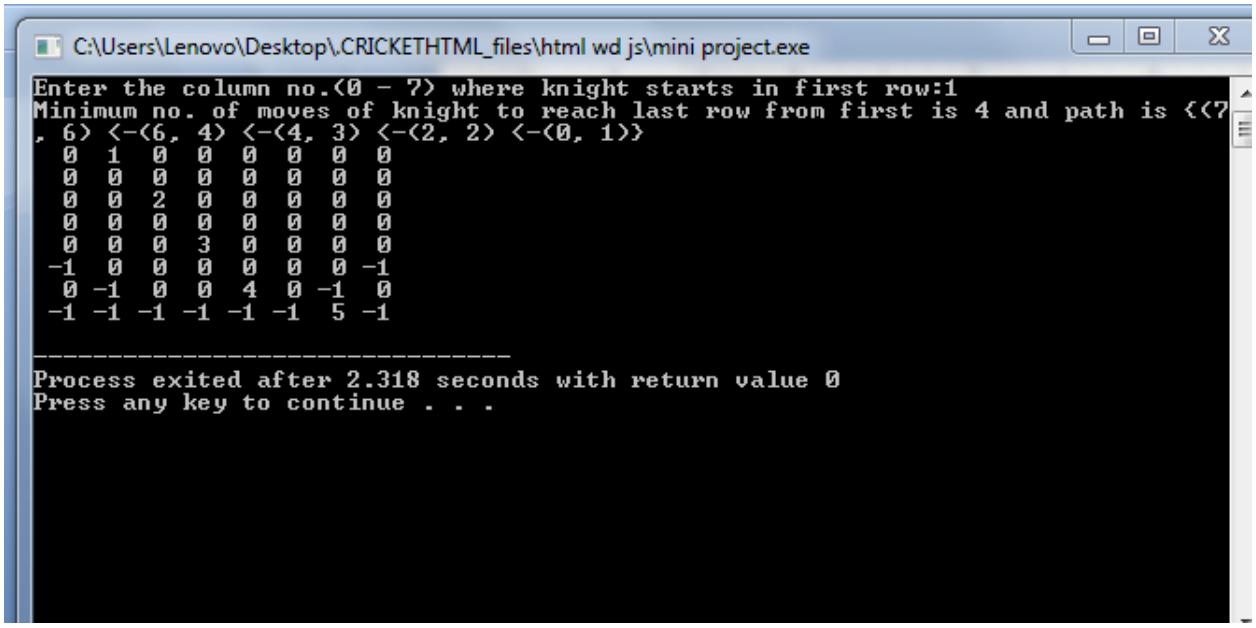
```
//     {
//         for (int j = 0; j < n; j++)
//         {
//             printf("%d ",board[i][j]);
//         }
//         printf("\n");
//     }


// }


//to track back the path taken by knight
void backtrack(int x, int y, int move){
    int dx[] = {-2, -1, 1, 2,  2,  1, -1, -2},
        dy[] = { 1,  2, 2, 1, -1, -2, -2, -1}, a, b;


    if (x != 0)
    {
        printf("(%d, %d) <-",x,y);
```

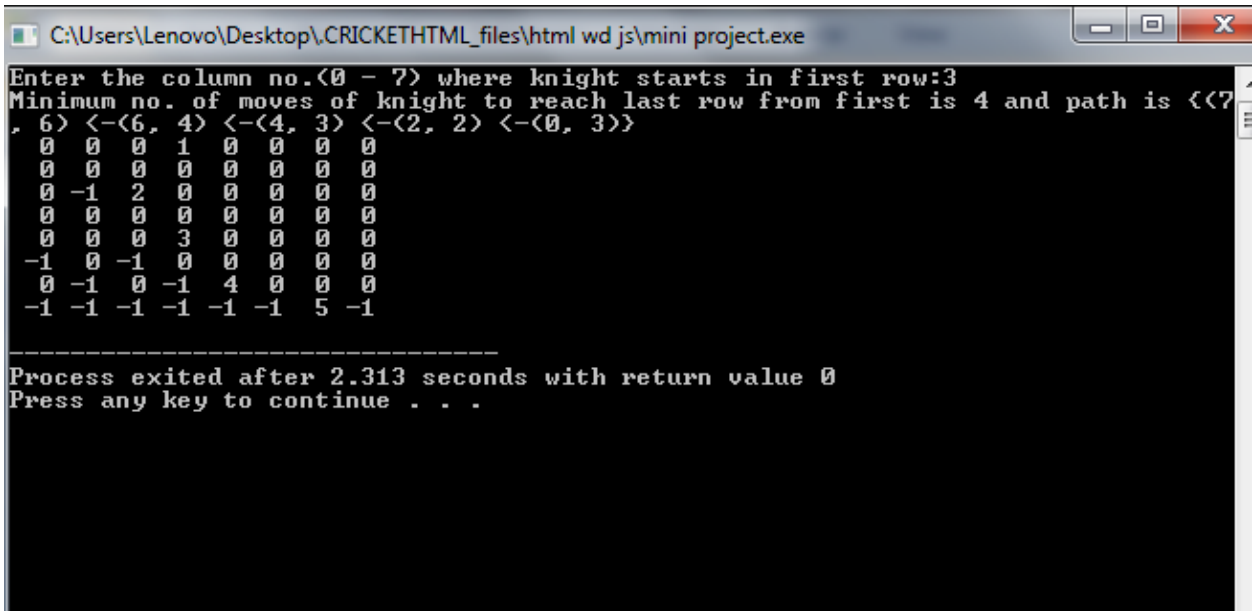
```
        visited[x][y]=move+1;
    }
    else{
        printf("(%d, %d)",x,y);
        visited[x][y]=move+1;
    }
    for (int i = 0; i < 8; i++)
    {
        a = x - dx[i] ; b = y - dy[i];
        if(isInside(a, b) && board[a][b] == move - 1){
            backtrack(a, b, move - 1);
            break;
        }
    }
}

int main()
{
```

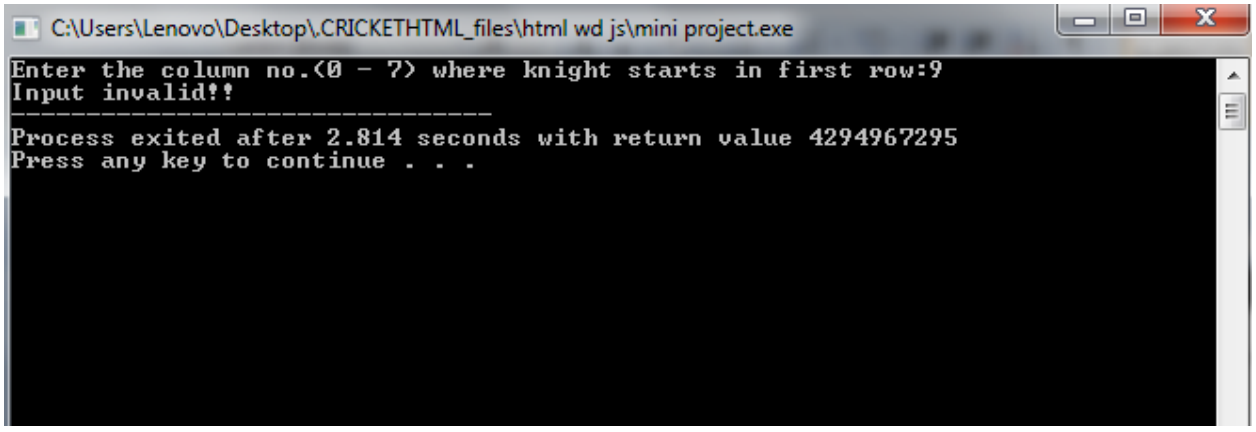
```
    int begCol,pos;
    printf("Enter the column no.(0 - 7) where knight starts in first row:");
    scanf("%d",&begCol);
    if(begCol >n - 1 || begCol < 0){
        printf("Input invalid!!");
        return -1;
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            visited[i][j] = -1;
            board[i][j] = -1;
        }
    }
    pos = steps(begCol);
    printf("Minimum no. of moves of knight to reach last row from first is %d and path is {",pos);
```

```
    backtrack(n-1, endCol, pos);
    printf("}\n");
    display();


    // displayB();
    return 0;
}
```

OUTPUT




C:\Users\Lenovo\Desktop\.CRICKETHTML_files\html wd js\mini project.exe
Enter the column no.(0 - 7) where knight starts in first row:3
Minimum no. of moves of knight to reach last row from first is 4 and path is ((7
, 6) <-(6, 4) <-(4, 3) <-(2, 2) <-(0, 3))
0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0
0 -1 2 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 3 0 0 0 0
-1 0 -1 0 0 0 0 0
0 -1 0 -1 4 0 0 0
-1 -1 -1 -1 -1 -1 5 -1
Process exited after 2.313 seconds with return value 0
Press any key to continue . . .


C:\Users\Lenovo\Desktop\.CRICKETHTML_files\html wd js\mini project.exe
Enter the column no.(0 - 7) where knight starts in first row:9
Input invalid!!
Process exited after 2.814 seconds with return value 4294967295
Press any key to continue . . .