**Q1:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

int counter(int number, int divider);

void main()

{

// Variables

int number, divider;

// Instructions and input

printf("Enter a number: ");

scanf("%d", &number);

printf("Enter a divider: ");

scanf("%d", &divider);

// Solution

int x = counter(number, divider);

printf("The number of divides without remainders: %d\n", x);

}

int counter(int number, int divider)

{

static int count = 0;

// Ending condition

if (number == 0)

{

return count;

}

// General progression

int temp = number % 10;

if (temp % divider == 0)

{

count++;

}

counter(number / 10, divider);

}

Graphical user interface, text

Description automatically generated

**Q2:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#define N 5

#include <stdio.h>

int path\_exists(int mat[][N], int rowdex, int coldex, int rows, int cols);

void main()

{

// Variables

int matrix[N][N] = { 0 }; // Coordinations map [NXN]

int rows = N - 1; // Rows number

int cols = N - 1; // Coloumns number

int rowdex = 0; // Vertical position

int coldex = 0; // Horizontal position

// Instructions and input

printf("Enter ones or zeroes for the %dX%d matrix.\n\n", N, N);

for (int i = 0; i < N; i++)

{

printf("For row #%d: ", i);

for (int j = 0; j < N; j++)

{

scanf("%d", &matrix[i][j]);

}

}

printf("\n");

// Function checking

int x = path\_exists(matrix, rowdex, coldex, rows, cols);

switch (x)

{

case 0: printf("The path does not exist\n"); break;

case 1: printf("The path exists\n"); break;

default: printf("There is an error in the code. The result of the function was neither 0 or 1.\n"); break;

}

}

int path\_exists(int mat[][N], int rowdex, int coldex, int rows, int cols)

{

// Success sign

static int trigger = 0;

// Success condition

if ((rows == 0) && (cols == 0) || (trigger == 1))

{

trigger = 1;

return trigger;

}

// Starting and ending conditions

if ((mat[rowdex][coldex] == 0) || (mat[N - 1][N - 1] == 0))

{

return 0;

}

// Border condition

if (rowdex > N - 1 || coldex > N - 1)

{

printf("There is an error in the code. The index exceeded the map's borders.\n");

return 0;

}

// General progression

// Diagonal step

if ((mat[rowdex + 1][coldex + 1] == 1) && ((rows > 0) && (cols > 0)))

{

path\_exists(mat, rowdex + 1, coldex + 1, rows - 1, cols - 1);

}

// Right step

if ((mat[rowdex][coldex + 1] == 1) && ((cols > 0)))

{

path\_exists(mat, rowdex, coldex + 1, rows, cols - 1);

}

// Down step

if ((mat[rowdex + 1][coldex] == 1) && ((rows > 0)))

{

path\_exists(mat, rowdex + 1, coldex, rows - 1, cols);

}

// Faliur condition

return trigger;

}

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

**Q3:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

int comparison(char s1[], char s2[]);

void main()

{

// Variables

char string1[30];

char string2[30];

// Instructions and input

printf("Enter two strings.\n");

printf("String 1: ");

gets(string1);

printf("String 2: ");

gets(string2);

// Function checking

int x = comparison(string1, string2);

switch (x)

{

case -1: printf("The first string is first in the order.\n"); break;

case 0: printf("The two strings are a complete match.\n"); break;

case 1: printf("The second string is the first in the order.\n"); break;

case 2: printf("Error #1 - the strings do not exist.\n"); break;

default: printf("An unknown error has occured in the function.\n"); break;

}

}

int comparison(char s1[], char s2[])

{

static int c = 0; // Position counter

// Error and match ending conditions

if ((s1[c] == '\0') && (s2[c] == '\0'))

{

// Error ending condition

if (c == 0)

{

return 2;

}

// Match ending condition

else

{

return 0;

}

}

// String 1 ending condition

if ((s1[c] > s2[c]) || ((s2[c] == '\0') && (s1[c] != '\0')))

{

return 1;

}

// String 2 ending condition

if ((s1[c] < s2[c]) || (s1[c] == '\0') && (s2[c] != '\0'))

{

return -1;

}

// General progression

if ((s1[c] == s2[c]) && (s1[c] != '\0') && (s2[c] != '\0'))

{

c++;

return comparison(s1, s2);

}

}

Graphical user interface, text

Description automatically generated with medium confidence

Text

Description automatically generated with medium confidence

Graphical user interface, text, application, chat or text message

Description automatically generated

**Q4:**

הפונקציה מחזירה את סכום הספרות של המספר שהוכנס אליה.