Question 1: Write a function to count number of occurrences of a character in a string

Code:

```c
// Write a function to count number of occurrences of a character in a string;

#include <stdio.h>

#include <stdio.h>

// function prototype
int countNumberOfOcurrences(char *s, char target);

int main()
{
    char *s = "Mehmood was here!"; //this is the input string
    char target = 'e'; // this is the target

    int count = countNumberOfOcurrences(s, target);

    printf("'%c' occurs %d times in the string \"%s\".\n", target, count, s);

    return 0;
}


// the function return the number of occurrences of target in string s
int countNumberOfOcurrences(char *s, char target)
{
    int count = 0; // initialzing the count to 0

    while (*s)  // traverse through each character of string
    {
        if (*s == target) // if the  current character matches target increment the count
        {
            count++;
        }
        s++;
    }

    return count; // return the final count
}
```

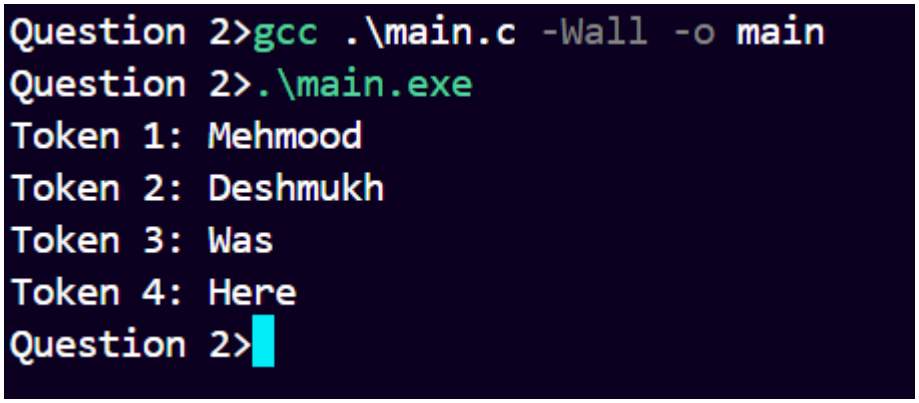
Output:

```
Question 1>gcc .\main.c -Wall -o main
Question 1>.\main.exe
'e' occurs 3 times in the string "Mehmood was here!".
Question 1>
```

Question 2: Write the strtok() function.

```c
// Write the strtok() function.

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

//function prototype
char* _strtok(char *s, char *delimiter);

int main() {
    char s[] = "Mehmood!Deshmukh,!Was,Here"; //this is the input string
    char *delimiter = ",!"; // this is the sequence of delimeters
    char *token;

    token = _strtok(s, delimiter);
    int index = 1;
    while (token != NULL) {
        printf("Token %d: %s\n", index++, token);
        token = _strtok(NULL, delimiter);
    }

    return 0;
}

// my function will accept multiple delimiters as a string and will work in all the cases: eg ",!;"
char* _strtok(char *s, char *delimiter) {
    static char *start = NULL; //static because we need to return the next token everytime
    char *token_start;

    // if s is NULL means we have to tokenize the same string we did previously. same as the strtok
function
    if (s != NULL) {
        start = s;
    }

    // if start is NULL which means there are no more tokens left so we will return NULL
    if (start == NULL) {
        return NULL;
    }

    //take care of leading delimiters
    while (*start) {
        char *d = delimiter; //compare each character with all the delimiters if any of them match
increment the start
        while (*d) {
            if (*start == *d) {
                ++start;
                break;
            }
            ++d;
        }
        if (*d == '\0') {
            break;
        }
    }

    // if we reached the end of the string we will return NULL;
    if (*start == '\0') {
        return NULL;
    }


    //else we will tokenize
    token_start = start;

    while (*start) {
        char *d = delimiter; // for each delimiter check if the current character matches any of them
        while (*d) {
            if (*start == *d) { //if the current character matches any of the delimiter , replace it with
null character and return the token
                *start = '\0';
                ++start;
                return token_start;
            }
            ++d;
        }
        ++start;
    }

    return token_start;
}
```

Output:


Question 2>gcc .\main.c -Wall -o main
Question 2>.\main.exe
Token 1: Mehmood
Token 2: Deshmukh
Token 3: Was
Token 4: Here
Question 2>

Question 3: Write a function which finds the longest possible subsequence of one string into another and returns the length + pointer to the subsequence

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

//this is a MACRO to get MAX of two values
#define MAX(a, b) ((a) > (b)) ? (a) : (b);

// function prototype
char* lcs(char* s1, char* s2, int* length);

int main() {
    char* s1 = "abcdemnopxyz"; //this is the string in which we will search for the longest subsequence
    char* s2 = "mnotq"; // we have to find the longest subsequence of this string

    int length; // this will store the length of the longest subsequence
    char* subsequence = lcs(s1, s2, &length); //get the LCS

    printf("The Longest Common Subsequence: %s\n", subsequence);
    printf("The Length of LCS: %d\n", length);

    free((void*)subsequence);

    return 0;
}

// initially i had written a function using recursion which was very inefficient and hence i took reference from the
internet where i got to know about the dunamic programming solution

//  here is the old solution:
// int lcs(char* s1, char* s2) {
//     if(*s1 == '\0' || *s2 == '\0') return 0;

//     if(*s1 == *s2) return 1 + lcs(s1 + 1, s2 + 1);

//     return MAX(lcs(s1 + 1, s2), lcs(s1, s2 + 1));

// }

// here is the dp solution
char* lcs(char* s1, char* s2, int* length) {
    int length1 = strlen(s1);
    int length2 = strlen(s2);

    int dp[length1 + 1][length2 + 1]; //initailize a 2d array

    for (int i = 0; i <= length1; ++i) {
        for (int j = 0; j <= length2; ++j) {
            if (i == 0 || j == 0) //if any of the strings length is 0 answer will be 0
                dp[i][j] = 0;
            else if (s1[i - 1] == s2[j - 1])
                dp[i][j] = 1 + dp[i - 1][j - 1];
            else
                dp[i][j] = MAX(dp[i - 1][j], dp[i][j - 1]);
        }
    }

    *length = dp[length1][length2];

    char* subsequence = (char*)malloc((*length + 1) * sizeof(char));

    int i = length1, j = length2, index = *length;
    while (i > 0 && j > 0) {
        if (s1[i - 1] == s2[j - 1]) {
            subsequence[index - 1] = s1[i - 1];
            i--;
            j--;
            index--;
        } else if (dp[i - 1][j] > dp[i][j - 1]) {
            i--;
        } else {
            j--;
        }
    }

    subsequence[*length] = '\0';

    return subsequence;
}
```

Output:

```
Question 3>gcc .\main.c -Wall -o main
Question 3>.\main.exe
The Longest Common Subsequence: mno
The Length of LCS: 3
Question 3>
```

Question 4: Write a function to find gcd() of 2 numbers. (See Dromey for this, after you have tried your bit)

```c
//  Write a function to find gcd() of 2 numbers. (See Dromey for this, after you have tried
// your bit)

#include <stdio.h>

//function prototype
int gcd(int a, int b);

int main() {
    int num1 = 16;
    int num2 = 12;

    printf("The greatest common divisor of %d and %d is %d", num1, num2, gcd(num1, num2));
    return 0;
}

//my solution
// int gcd(int a, int b){
//     if(b == 0) return a;

//     return gcd(b, a % b);
// }

// dromey's solution
int gcd(int a, int b){
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```

Output:

```
Question 4>gcc .\main.c -Wall -o main
Question 4>.\main.exe
The greatest common divisor of 16 and 12 is 4
Question 4>
```

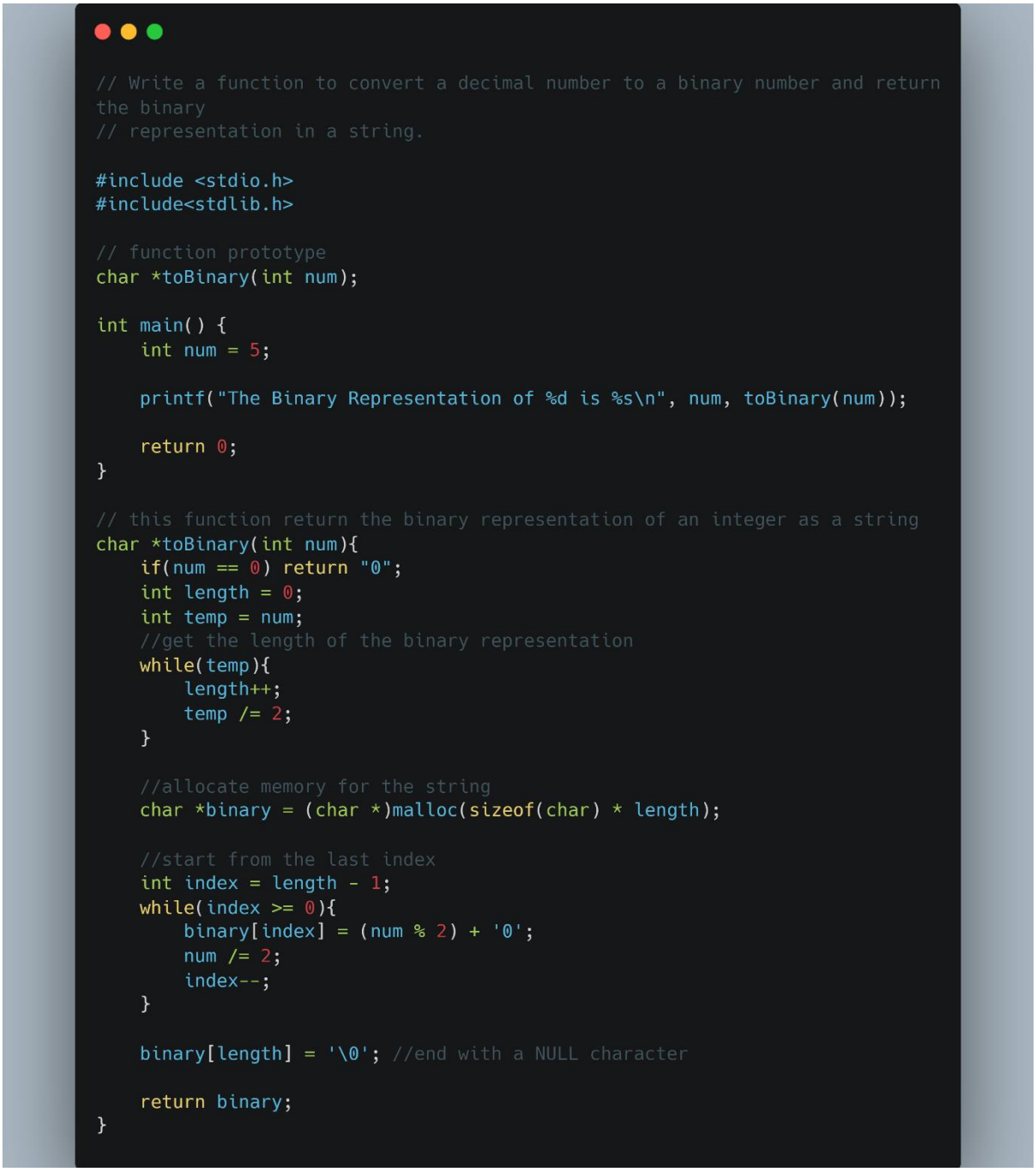
Question 5: Write a function to find lcm() of 2 numbers

```c
//  Write a function to find lcm() of 2 numbers

#include <stdio.h>

//function prototypes
int gcd(int a, int b);
int lcm(int a, int b);

int main() {
    int num1 = 16;
    int num2 = 12;

    printf("The lowest common multiple of %d and %d is %d", num1, num2, lcm(num1, num2));
    return 0;
}

//function for calculating the GCD
int gcd(int a, int b){
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

// using the formula LCM = a * b / GCD
int lcm(int a, int b){
    return a * b / gcd(a, b);
}
```

Output:

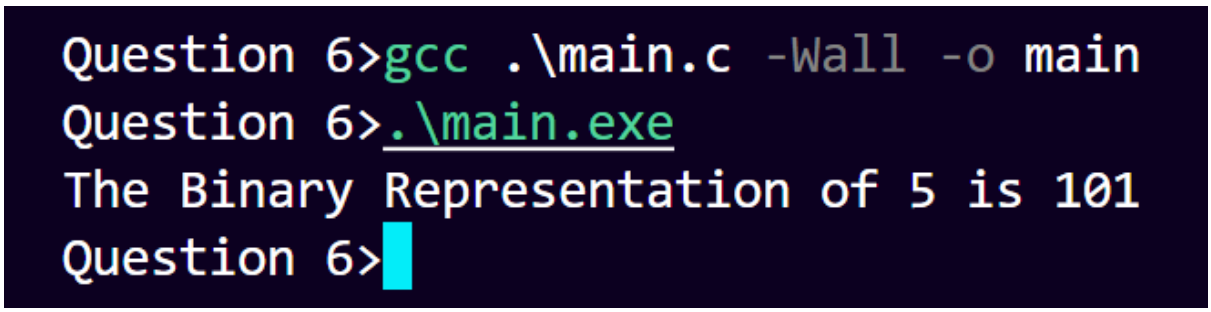
```
Question 5>gcc .\main.c -Wall -o main
Question 5>.\main.exe
The lowest common multiple of 16 and 12 is 48
Question 5>
```

Question 6: Write a function to convert a decimal number to a binary number and return the binary representation in a string.



```c
// Write a function to convert a decimal number to a binary number and return
the binary
// representation in a string.

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

// function prototype
char *toBinary(int num);

int main() {
    int num = 5;

    printf("The Binary Representation of %d is %s\n", num, toBinary(num));

    return 0;
}

// this function return the binary representation of an integer as a string
char *toBinary(int num){
    if(num == 0) return "0";
    int length = 0;
    int temp = num;
    //get the length of the binary representation
    while(temp){
        length++;
        temp /= 2;
    }

    //allocate memory for the string
    char *binary = (char *)malloc(sizeof(char) * length);

    //start from the last index
    int index = length - 1;
    while(index >= 0){
        binary[index] = (num % 2) + '0';
        num /= 2;
        index--;
    }

    binary[length] = '\0'; //end with a NULL character

    return binary;
}
```

Output:



```
Question 6>gcc .\main.c -Wall -o main
Question 6>.\main.exe
The Binary Representation of 5 is 101
Question 6>
```

Question 7: Write your own code for following library functions:
strcasecmp strsep strcasecmp strcoll

```c
// Write your own code for following library functions:
// strcasecmp strsep strcasecmp strcoll

#include <stdio.h>
#include <string.h>
#include <ctype.h>

// function prototypes for starcasecmp and strsep
int _strcasecmp(char *s1, char *s2);
char *_strsep(char **s, char *delimiters);

int main() {
    char str1[] = "Mehmood";
    char str2[] = "mehmood";

    int result = strcasecmp(str1, str2);

    if (result == 0) {
        printf("'%s' and '%s' are the same (ignoring case).\n", str1, str2);
    } else if (result < 0) {
        printf("'%s' is less than '%s' (ignoring case).\n", str1, str2);
    } else {
        printf("'%s' is greater than '%s' (ignoring case).\n", str1, str2);
    }

    char str[] = "apple,orange,,banana,grape";
    char *token;
    char *stringp = str;

    while ((token = _strsep(&stringp, ",")) != NULL) {
        printf("Token: '%s'\n", token);
    }

    return 0;
}

// this function compares two strings case insensitively
int _strcasecmp(char *s1, char *s2){
    //skip same characters
    while(tolower(*s1) == tolower(*s2)){
        if(!*s1) return 0;

        s1++;
        s2++;
    }

    // return difference between the first different character
    return tolower(*s1) - tolower(*s2);
}

//similar to strtok but it returns the first token it finds and it also
modifies the input string
char *_strsep(char **s, char *delimiters) {
    char *start = *s;
    char *temp;

    if (start == NULL) {
        return NULL;
    }

    // for each delimiter check if current character matches it.
    // if it matches replace it with '\0'
    //set the string to start from the next character
    for (temp = start; *temp != '\0'; temp++) {
        for (char *d = delimiters; *d != '\0'; d++) {
            if (*temp == *d) {
                *temp = '\0';
                *s = temp + 1;
                return start;
            }
        }
    }

    *s = NULL;
    return start;
}
```

Output:

```
Question 7>gcc .\main.c -Wall -o main
Question 7>.\main.exe
'Mehmood' and 'mehmood' are the same (ignoring case).
Token: 'apple'
Token: 'orange'
Token: ''
Token: 'banana'
Token: 'grape'
Question 7>
```

Question 8: What are wide characters? Write a sample code using wcscmp function. (You have to read wide character strings and then call wcscmp function)

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

// Wide characters are a form of character representation about processing a more comprehensive
set of characters than the standard single-byte character, usually  char  in C. They come in
very handy during the processes of internationalization and providing support to different
languages and symbols that cannot be represented using one byte only.

// Characteristics of Wide Characters :
// Size: The wide characters are mostly represented by more than one byte. The size usually
varies depending on a system and specific type of wide character used. Example:
//
// In most of the implementations, wchar_t– the most commonly used wide character type– is 2
bytes on some systems, for example, Windows, and it is 4 bytes on other systems, like most of
the Unix-like systems.
// How wide character is encoding: The encoding of wide character may vary. Example:
// On Windows, wchar_t is mostly, encoded as UTF-16.
// In most Unix-like systems, wchar_t is usually encoded in UTF-32.
// Standard Library Support: The standard library of C supports wide characters through several
functions and types defined in <wchar.h>, for example:
// Type wchar_t to represent wide characters.
// Functions wprintf(), wscanf(), wcslen(), wcscmp() etc. for I/O and manipulation of wide
characters.
// Use Cases:
// These are used to deal with languages/symbols where more than one byte is required per
character.
// They are very useful in rendering great assistance to Unicode since it contains many
characters and symbols from most of the written alphabets.

int main() {
    wchar_t str1[] = L"Hello";
    wchar_t str2[] = L"World";

    int result = wcscmp(str1, str2);

    if (result == 0) {
        wprintf(L"The strings are equal.\n");
    } else if (result < 0) {
        wprintf(L"The first string is less than the second string.\n");
    } else {
        wprintf(L"The first string is greater than the second string.\n");
    }

    return 0;
}
```

Output:

```
Question 8>gcc .\main.c -Wall -o main
Question 8>.\main.exe
The first string is less than the second string.
Question 8>
```

Question 9: Write following functions: sine,sine-inverse,cosine, cosine-inverse.Then using sine and cosine, write tan() function. Check whether calling sine(sineinverse(x)) on your own functions gives you X

```c
//rite following functions: sine, sine-inverse, cosine, cosine-inverse. Then using sine and
// cosine, write tan() function. Check whether calling sine(sine-inverse(x)) on your own
// functions gives you x.

#include <stdio.h>
#include <math.h>

// instead of using direct values its better to define macros
#define PI 3.14159265358979323846
#define EPSILON 1e-10

// function to normalize the angle , used in sine and cosine calculation
double normalize_angle(double x) {
    while (x > PI) x -= 2 * PI;
    while (x < -PI) x += 2 * PI;
    return x;
}

//function which uses the taylor series to calculate sine
double sine(double x) {
    x = normalize_angle(x);
    double term = x;
    double sum = x;
    int n = 3;

    while (fabs(term) > EPSILON) {
        term *= -x * x / ((n) * (n - 1));
        sum += term;
        n += 2;
    }

    return sum;
}

//function which uses the taylor series to calculate cosine
double cosine(double x) {
    x = normalize_angle(x);
    double term = 1;
    double sum = 1;
    int n = 2;

    while (fabs(term) > EPSILON) {
        term *= -x * x / ((n) * (n - 1));
        sum += term;
        n += 2;
    }

    return sum;
}

//function which uses the taylor series to calculate sine-inverse
double sine_inverse(double x) {
    double term = x;
    double sum = x;
    int n = 1;

    while (fabs(term) > EPSILON) {
        term *= x * x * (2 * n - 1) * (2 * n - 1) / ((2 * n + 1) * (2 * n));
        sum += term;
        n++;
    }

    return sum;
}
```

```c
// function which uses the formula: cos^-1(x) = PI/2 - sin^-1(x)
double cosine_inverse(double x) {
    return PI / 2 - sine_inverse(x);
}

// function which uses the formula: tan(x) = sin(X) / cos(x)
double tan(double x) {
    return sine(x) / cosine(x);
}

int main() {
    double x = PI / 4;

    double sine_value = sine(x);
    double cosine_value = cosine(x);
    double tan_value = tan(x);

    printf("x = %f\n", x);
    printf("sin(x) = %f\n", sine_value);
    printf("cos(x) = %f\n", cosine_value);
    printf("tan(x) = %f\n", tan_value);

    double sine_inverse_value = sine_inverse(sine_value);
    double cosine_inverse_value = cosine_inverse(cosine_value);

    printf("sin^-1(sin(x)) = %f\n", sine_inverse_value);
    printf("cos^-1(cos(x)) = %f\n", cosine_inverse_value);

    return 0;
}
```

Output:

```
Question 9>gcc .\main.c -Wall -o main
Question 9>.\main.exe
x = 0.785398
sin(x) = 0.707107
cos(x) = 0.707107
tan(x) = 1.000000
sin^-1(sin(x)) = 0.785398
cos^-1(cos(x)) = 0.785398
```

Question 10: Write a program to reverse the digits of an integer and store the result as another integer.

```c
// Write a program to reverse the digits of an integer and store the result as another
// integer.

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

//function prototype
int reverse(int num);

int main( ) {
    int num = 12345;

    printf("The Reverse Representation of %d is %d\n", num, reverse(num));

    return 0;
}

//function to reverse digits of an integer
int reverse(int num){
    int result = 0;

    while(num){
        int digit = num % 10;
        result = result * 10 + digit;
        num /= 10;
    }

    return result;
}
```

Output:

```
Question 10>gcc .\main.c -Wall -o main
Question 10>.\main.exe
The Reverse Representation of 12345 is 54321
Question 10>
```

Question 11: Write a program which reads a string and if the string has all digits in it, then derives the integer it represents

```c
//Write a program which reads a string and if the string has all digits in it, then
// derives the integer it represents
#include <stdio.h>
#include <ctype.h>
#include <limits.h>

#define MAX_SIZE 1024

//function to derive integer from a string if it contains all numbers
int deriveInteger(const char *s) {
    int result = 0;

    while (*s != '\0') {
        if (isdigit(*s)) {
            result = result * 10 + (*s - '0');
        } else {
            return INT_MIN; // if not a digit return invalid
        }

        s++;
    }

    return result;
}

int main() {
    char input[MAX_SIZE];

    printf("Enter a string represeting digits: ");
    scanf("%s", input);

    int result = deriveInteger(input);

    if (result != INT_MIN) {
        printf("Derived integer: %d\n", result);
    } else {
        printf("The string does not represent a valid integer!\n");
    }

    return 0;
}
```

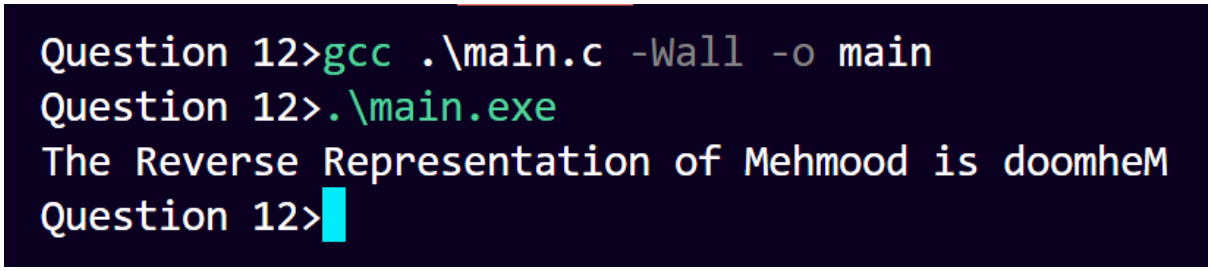
Output:

```
Question 11>gcc .\main.c -Wall -o main
Question 11>.\main.exe
Enter a string represeting digits: 12345
Derived integer: 12345
Question 11>.\main.exe
Enter a string represeting digits: 1234a
The string does not represent a valid integer!
```

Question 12: Write a function which does the following void rev(char *str); Reverses the string "str" in place (without using another string).

```c
// Write a function which does the following
// void rev(char *str);
// Reverses the string "str" in place (without using another string).

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

//function prototype
void reverseString(char *str);

int main() {
    char str[] = "Mehmood";

    printf("The Reverse Representation of %s is ", str);
    reverseString(str);
    printf("%s\n", str);

    return 0;
}

//function to reverse string in place
void reverseString(char *str){
    int len = strlen(str);

    for(int i = 0; i < len / 2; i++){
        char temp = str[i];
        str[i] = str[len - 1 - i];
        str[len - 1 - i] = temp;
    }
}
```

Output:



```
Question 12>gcc .\main.c -Wall -o main
Question 12>.\main.exe
The Reverse Representation of Mehmood is doomheM
Question 12>
```

Question 13: Write a function which cuts a string given by "str" on the character given in "ch" and returns the first such word. char *cutonchar(char *str, char ch); For example: if 'str' is "something bad" and 'ch' is ' ' then it returns "something". if 'str' is "something bad" and 'ch' is 'e' then it returns 'som'

```c
//Write a function which cuts a string given by "str" on the character given in "ch"
// and returns the first such word.
// char *cutonchar(char *str, char ch);
// For example: if 'str' is "something bad" and 'ch' is ' ' then it returns "something".
// if 'str' is "something bad" and 'ch' is 'e' then it returns 'som'

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

//function to cut string on character
char* cutonchar(char* str, char ch) {
    char* result = (char*)malloc(strlen(str) + 1);

    if (result == NULL) {
        return NULL;
    }

    strcpy(result, str);
    char* pos = strchr(result, ch); //get position of ch

    if (pos != NULL) { //if ch exixts replace it with '\0'
        *pos = '\0';
    }

    return result;
}

int main() {
    char input[] = "I am Mehmood Rehan Deshmukh";
    char ch = 'o';
    printf("%s cut on %c is %s", input, ch, cutonchar(input, ch));
    return 0;
}
```

Output:

```
Question 13>gcc .\main.c -Wall -o main
Question 13>.\main.exe
I am Mehmood Rehan Deshmukh cut on o is I am Mehm
Question 13>
```