Assignment 1: C calling convention, Decimal to Hexadecimal

Submissions which deviate from the instructions will not be graded!

Please make your output as in the examples below, except for the leading zeros in the output of task 2, which you are allowed to have.
Otherwise automatic scripts might fail on your assignment and you will lose your points for this.

Assignment Description

This assignment contains two tasks:

Task 1 - C calling convention

You will implement entirely on your own (unlike task 2, we provide no files for this task). You may use C library functions fgets, printf, sscanf. Your program should be composed of two files, one written in C and one written in assembly language, according to the instructions below.

Write a C program that contains all your C code: function 'main(…)' performs the following steps:

1. Prompts for and reads one integer (32 bits) number x in decimal base from the user (you may use fgets() and sscanf()).
2. You may assume that the input is always valid.
3. Calls 'void assFunc(int x)' written in assembly language with the above integer as an argument

'void assFunc(int x)' performs the following steps:

1. Calls 'char c_checkValidity(int x)' written in C to check the number x (as shown bellow)
2. If c_checkValidity(int x) returns 1, calculate `z=x*4`, and print z in decimal base, using printf with '%d' format
3. Otherwise, calculate `z=x*8`, and prints z in decimal base, using printf with '%d' format

Note: You are not allowed to use the arithmetic instructions ADD, SUB, MUL and similar to compute the results, all these computations are to be done using bitwise logical and shift instructions.

'char c_checkValidity(int x)' performs the following:

1. Returns 1 if x is even
2. Otherwise, returns 0.

Examples:

```
> task1Assignment1.out
5
40


> task1Assignment1.out
8
32
```

Task 2 - Converting unsigned numbers in hexadecimal base to unsigned number in binary base

We provide you a simple program in C that inputs a string (as a line) from the user, and calls the convertor(…) function **that you need to implement in assembly language**. You need to change it into a loop which is terminated when user enters the "q" input string.

The input string contains ASCII digits, representing an unsigned hexadecimal number

- convertor(…) receives a pointer to the beginning of a null terminated string.
- You may assume that the input string contains only decimal digits (i.e. '0' , '1' , … , '9'), and the letters A, B, C, D, E and F or only 'q' in a case that a user wants to quit the program
- The output string should also be null terminated.
- Note that the input string may contain a newline ('\n') character, make sure you are aware of it and treat it accordingly.
- convertor(…) should print the resulting binary representation as a **null terminated string**. That is, in your assembly code you should call the printf function with the '%s' format.

Note: You are not allowed to perform multiplication and/or division between numbers.

Examples:

```
> task2Assignment1.out
1DF
111011111
9F1
100111110001
0
0
q
```

Character conversion will **not** be in place this time.
You should write the output string into another buffer (provided in the code skeleton) before printing.

Note: you **may not** use any available function that automatically does the conversion, such as printf, to do the conversion. You need to compute the output digits yourself.