In this lab, we will write 4 functions to convert between strings of '0's and '1's and binary numbers. For each function, write down the corresponding test code that prompts the user for a value. You can assume that the user input is correct. Numbers are 32 bits wide. Functions should comply with MIPS calling conventions (parameters in $a registers, return value in $v registers, saved registers ...).

1) Unsigned conversion from string to binary number

Write a function that takes a string of '0's and '1's and returns the corresponding unsigned binary number.

Function Definition:
   unsigned strtobin u(char *str)

   Parameters:
       str: address of the string

   Return value:
       The corresponding unsigned binary number

Example:
   str: "101000100"
   Return value: the number 00000000000000000000000101000100b (0x00000144 or decimal 324). Notice how the string gets “zero extended”.
2) Signed conversion from string to binary number

Write a function that takes a string of '0's and '1's and returns the corresponding **signed** binary number.

Function Definition:
```c
int strtobin (char *str)
```

Parameters:
- `str`: address of the string

Return value:
- The corresponding **signed** binary number

Example 1:
- `str: "101000100"
- Return value: the number 0xFFFFFF44 or decimal -188. Notice how the string gets “sign extended” with '1's.

Example 2:
- `str: "0101000100"
- Return value: the number 0x00000144 or decimal 324. Notice how the string gets “sign extended” with '0's.

3) Unsigned conversion from binary number to string

Write a function that takes an **unsigned** binary number and converts it into a string of '0's and '1's.

Function Definition:
```c
void bintostru (unsigned value, char *str):
```

Parameters:
- `value`: the value to store as a string
- `str`: address of the string buffer

Return value:
- None

Example:
- `value: 0x00000144`
- Output: `str` should now contain "101000100". Notice how leading '0's are trimmed.
4) Signed conversion from binary number to string

Write a function that takes a signed binary number and converts it into a string of '0's and '1's.

Function Definition:

    void bintostr (int value, char *str):

Parameters:
    value: the value to store as a string
    str: address of the string buffer

Return value:
    None

Example 1:
    value: 0x00000144
    Output: str should now contain "00000000000000000000000101000100" . Notice how leading '0's are not trimmed.

Example 2:
    value: 0xFFFFF44
    Output: str should now contain "1111111111111111111111101000100" . Notice how leading '1's are not trimmed.