Homework 1

Due: April 18, 2024 **Points:** 100

All programs are to be submitted through Gradescope. Gradescope will compile and execute your program, sometimes with varying inputs or invocations. You can see your score once executions are done. You can submit your program multiple times, up to the deadline — at that point, the score you have will *usually* be the grade for that problem. I say "usually" because we reserve the right to look at your submission and deduct points if you do not use proper programming style.

You have to name your program as given in the problem. Otherwise you will get an odd error message indicating there is a problem with Gradescope.

Also, your output must match Gradescope's *exactly*, including blanks and tabs — so pay attention to the example output!

1. (*15 points*) Please write a small program that outputs a tic-tac-toe board. Your output is to look exactly like this:

```
       #       #
       #       #
       #       #
       #       #
       #       #
######################
       #       #
       #       #
       #       #
       #       #
       #       #
######################
       #       #
       #       #
       #       #
       #       #
       #       #
```

   When this is done, exit with the status code 96 (*not* 0).

   Call your program "ttt.c".

2. (*25 points*) The program *change.c*, available on Canvas, prints the coins that are needed to make change of a specified amount. Currently, you need to set the variable `amount` to the value you want in change. That is inconvenient, to say the least.

   This programming assignment is to change the program to request a value, and read it from the standard input. Be sure to check that the value given is a nonnegative integer!

   Your output must match Gradescope's output ***exactly***. Here are some examples of inputs and outputs. Input is shown in red, and "␣" represents a newline. Note that there is a prompt for the input!

```
Amount? 99␣
3 quarters, 2 dimes, 0 nickels, 4 pennies

Amount? 42␣
1 quarters, 1 dimes, 1 nickels, 2 pennies

Amount? hello␣
Please enter a non-negative integer
```

The last one prompts on the standard output, but prints the error message on the standard error, not the standard output.

Call your program "change.c".

Hint: Use `scanf` to read the input.

3. (*30 points*) You are to write a loop that print numbers in sequence. The sequence may be increasing or decreasing. The file *loopy.c*, available on Canvas, contains a wrapper for this loop. Look at the function `loop`. It takes 2 arguments: `m`, the starting point and `n`, the ending point. Here are the cases:

   If $m < n$, then print $m, m+1, \ldots, n$, each number on a separate line.
   If $m = n$, then just print m.
   If $m > n$, then print $m, m-1, \ldots, n$, each number on a separate line.

   Again, because your output must match Gradescope's output ***exactly***, here are some examples. When you execute the program, called *loopy*, give the starting point as the first argument and the ending point as the second argument. Again, what you type is shown in red, and "␣" represents a newline.

```
loopy 1 10␣
1
2
3
4
5
6
7
8
9
10

loopy -3 -10␣
-3
-4
-5
-6
-7
-8
-9
-10

Amount? loopy 0 0␣
0
```

   Call your program "loopy.c".

4. (*30 points*) The program *show.c*, available on Canvas, is supposed to read characters from the standard input and print them on the standard output after expanding any non-printing characters to their C character escape sequence.

   The relevant characters, and the C escape sequences to be printed when those characters are encountered, are:

character	print as	character	print as
newline	\n	backslash	\\
horizontal tab	\t	vertical tab	\v
backspace	\b	carriage return	\r
form feed	\f	bell	\a
NUL	\0	*anything else*	\x*xx*

The "anything else" entry means that any non-printing character other than the ones named in the table is to be printed as a sequence of 2 hexadecimal digits preceded by a backslash and an "x" (so, for example, the character SOH prints as "\x01"). When the escape sequence for a newline is printed, the program is to skip to the next line.

Unfortunately, the program as saved in *show.c* will not even compile, let alone run. And the programmer thoughtlessly left off all the comments. Hence, your mission: fix the program so it works as described above. You are to turn in a corrected source program, with comments describing the changes you made to get it to work, to Gradescope, which will verify it works correctly.

Call your fixed program *show.c*.