cs201 – sum 2020

Assignment 2 – (40 points)

Download the assign2a.c and assign2b.c files from the assignment 2 folder on d2l. Do the following:

Part A: rounding floating point(20 points)

In the given file round.c, write four functions that take a float and round this value according to one of the four floating point rounding modes. The float can be either positive or negative. You can use the **fabs** and **floor** function in your code but no other math functions.

```
----------------------
EXAMPLE RUN #1:
----------------------
Enter float number:
2.5
Round to nearest even: 2.000000
Round to minus infinity: 2.000000
Round to plus infinity: 3.000000
Round toward zero: 1.000000
----------------------
EXAMPLE RUN #2:
----------------------
Enter float number:
3.5
Round to nearest even: 4.000000
Round to minus infinity: 3.000000
Round to plus infinity: 4.000000
Round toward zero: 2.000000
----------------------
EXAMPLE RUN #3:
----------------------
Enter float number:
-2.3
Round to nearest even: -2.000000
Round to minus infinity: -3.000000
Round to plus infinity: -2.000000
Round toward zero: -2.000000
----------------------
EXAMPLE RUN #4:
----------------------
Enter float number:
-3.5
Round to nearest even: -4.000000
Round to minus infinity: -4.000000
Round to plus infinity: -3.000000
Round toward zero: -3.000000
----------------------
```

Part B: Using extended gcc inline assembly (20 points)

NOTE: Do this part after studying week 4 material

NOTE: see the website:

http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html#s4

for how to do gcc inline assemby.

In file payroll.c, there are two functions you need to fill in. In the function **regular_pay_asm()**, write the assembly code needed to calculate regular pay.

Regular pay is defined as pay rate multiply by the number of hours equal to or under 40. So the algorithm would be:

if (hours > 40)

hours = 40

pay = hours X payrate

In the function **regular_pay_C()** write the C equivalent to the above algorithm in C code. This should be written first so you can compare the output of the asm code to the C code.

```
----------------------
EXAMPLE RUN #1:
----------------------
enter pay rate: 21
enter hours: 30
regular pay (C) is: 630
regular pay (asm) is: 630
----------------------
EXAMPLE RUN #2:
----------------------
enter pay rate: 21
enter hours: 54
regular pay (C) is: 840
regular pay (asm) is: 840
----------------------
EXAMPLE RUN #3:
----------------------
enter pay rate: 21
enter hours: 40
regular pay (C) is: 840
regular pay (asm) is: 840
----------------------
```

Extra Credit (15 points): Extend Part B Code

Note: This is extra credit and not necessary to get full points for the assignment

In the function "overtime_pay_asm", write the assembly code needed to calculate overtime pay. Overtime pay is defined as pay rate multiply by the number of hours over 40. If the hours are under 40; this value is 0. So the algorithm would be:

```
if (hours <= 40)
    hours = 0
else
    hours = hours - 40
pay = hours X payrate
```

In the function "overtime_pay_C" write the C equivalent to the above algorithm in C code. This should be written first so you can compare the output of the asm code to the C code.

```
----------------------
EXAMPLE RUN #1:
----------------------
enter pay rate: 21
enter hours: 30
regular pay (C) is: 630
regular pay (asm) is: 630
overtime pay (C) is: 0
overtime pay (asm) is: 0
total gross pay: 630
----------------------
EXAMPLE RUN #2:
----------------------
enter pay rate: 21
enter hours: 54
regular pay (C) is: 840
regular pay (asm) is: 840
overtime pay (C) is: 294
overtime pay (asm) is: 294
total gross pay: 1134
----------------------
EXAMPLE RUN #3:
----------------------
enter pay rate: 21
enter hours: 40
regular pay (C) is: 840
regular pay (asm) is: 840
overtime pay (C) is: 0
```

```
overtime pay (asm) is: 0
total gross pay: 840
----------------------
```