Name: Riley Payung

CDS 251 Final

Due May 14, 2020 by 11:59 pm on Blackboard

Exam is 115 out of 100 possible points. (15 points built in Extra Credit)

On your Honor!

- Complete this exam in a single sitting.
- Open notes but **no collaboration, no internet.**
- Do binary by hand, you may use a calculator for decimal calculations but no 'computer science' calculators (or google) to do binary operations or conversions for you.
- Exam is designed for a 2 hr period. You may take as long as you like as long as you do the exam in a **single sitting.**
- After you complete the exam, you will not return to it.
- Do not discuss the exam with anyone until after the due date.
- Fill in the information below after completing the exam.

Date exam taken: 5/13/2020

Time spent on exam: hour min 1:27

I attest that I have adhered to the above guidelines:

Your signature

Name: Riley Payung

CDS 251 Final Exam

1) (5 pts) Add these two **unsigned** bytes. Either convert to decimal or explain the error in the addition.

	128	64	32	16	8	4	2	1
	1	0	1	0	1	1	1	0
+	0	0	1	1	1	1	0	1
	1	1	1	0	1	0	1	1

128 + 64 + 32 + 8 + 2 + 1 = 235

2) (5 pts) Add these two **signed** bytes. Either convert to decimal or explain the error in the addition.

	128	64	32	16	8	4	2	1
	1	1	1	1	1			
	1	1	1	0	1	1	1	0
+	0	1	0	1	1	1	0	0
	0	1	0	0	1	0	1	0

- 64 + 8 + 2 + 0

-74

3) (10 pts) The following is a four byte Single Precision Real Number as stored in computer memory. What is the number in decimal?

1 10000000 01000000000000000000000

- .0+1/4+0...

$-\left(1 + \text{an. } \frac{1}{4}\right)^{\text{exp.}} = -1.25$

128 64 32 16 8 4 2 1
1 0 0 0 0 0 0 0

128 - 127 = 1

4) (5 pts) Write down a Fortran line that updates the variable 'Average' for computing the average of numbers you are reading in. Just one line, correct Fortran syntax.

read into x.

Avg = Avg + float(x/n)

Name:_____

5) (5 pts) Write a Fortran variable declaration that declares a two dimensional real array to store the state of 400,000 planets, where each planet can have x, y, and z coordinates and Vx, Vy, Vy velocities all in this array. The dimensions are ordered such that it minimizes computer cache misses. (Fixed size array here, do not use allocatable memory.)

real :: Planets(6, 400000)

6) (5 pts) How many bytes away from each other are the x-coordinates of two planets that are next to each other in the array? Assume the array is of type **real**.

6 x 4 bytes = 24 bytes apart.

7) (5 pts) In Big $O()$ notation, what are the computational complexities of Bubble Sort and Quick Sort?

Bubble is $O(N^2)$

Quick is ~~$O(\log(N) \cdot N)$~~ $O(N \log(N))$

8) (10 pts) QuickSort takes 2.1 seconds to sort 60,000 numbers. Bubble Sort takes 1.1 minutes. How long would it take to do 12 million numbers with each algorithm? Answer in appropriate units (one that a normal human does not have to stop and think how long this is).

Bubble:

60000 numbers in 66 seconds

$220^2 \times 66 = 2{,}904{,}000$ seconds

33.61 days.

66 seconds to do bubble

$\frac{12000000}{60000} = 200$

200×1.1 minutes

$\frac{60000}{66}$ = 909.09 per second

12m x 909.09

10,909,090,909 seconds

345,925 years

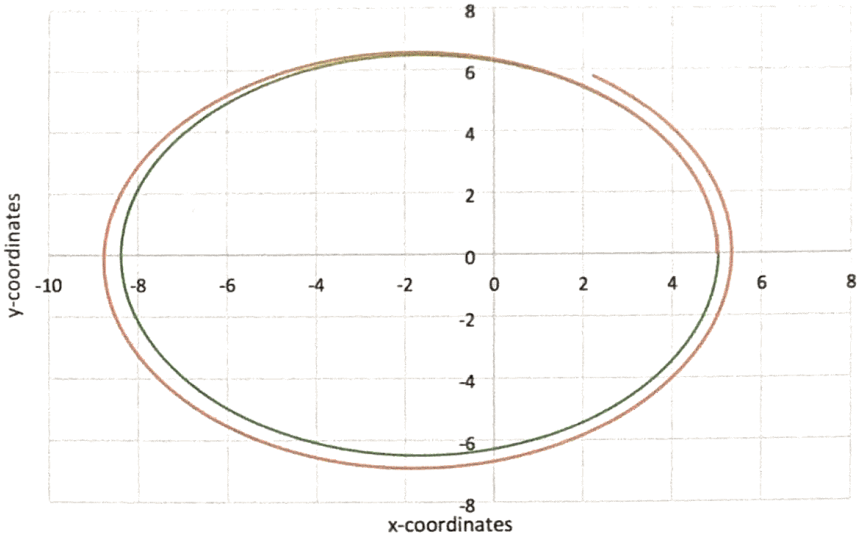
Quick Sort:

$66 \times \frac{12000000 \times \log_2(12000000)}{60000 \times \log_2(60000)} = 1957$ seconds

$= 5.43$ hours

=

Name:_____



9) (5 pts) The planet simulation with Forward Euler numerical solution produced the above two orbits. What kind of error dominates numerical simulations of differential equations? Circle One:

Rounding Error : Precision Error : (Truncation Error)

10) (5 pts) In the above orbits, one simulation was done with time step $h = 0.01$ and one was done with time step $h = 0.1$. Which one is which (Orange/Green)?

Orange: $h = 0.01$ Green: $h = 0.1$

11) (5 pts) Rewrite this equation such that it is set up to be programmed as a fixed point method. Clearly indicate the equation $g(x)$.

$$2x^2 - 4x + 2 = 0$$

$g(x) = \frac{2x^2 + 2}{4}$

$2x^2 = 4x - 2$

$x^2 = 2x - 1$

$x = \sqrt{2x-1}$

~~g(x) = (2 * x ** 2) (4 * x) + 2~~

$g(x) = \frac{x^2+1}{2}$

$g(x) = \sqrt{2x-1}$

12) (5 pts) There is a root 'close' to 2. Pick one of your two equations from problem 11 and determine if you would converge on a root if your initial guess is 2. Use the convergence criterion.

$\frac{d}{dx}\left(\frac{x^2}{2} + \frac{1}{2}\right)$

$= 2x + 0$

$\frac{d}{dx}\left(\sqrt{2x+1}\right) = \frac{1}{2}(2x+1)^{\frac{-1}{2}} = \frac{1}{2(2x+1)^{1/2}}$

$= \frac{1}{2\sqrt{2x+1}}$

The second one converges at 2. since we have no value (a holes) at 2.

Name:_____

13) (10 pts) Write the do loop for Newton's Method for root finding for the following equation. Do not write separate Fortran functions, just put the math in the do loop.

$$f(x) = x^2 + \cos(x) = 0$$

$$f'(x) = 2x - \sin(x) = 0$$

```
do
  if (i.eq. 50) exit
  if ( myfunc(x) .le. threshold) exit
  x = x - (myfunc(x) / myfunc2(x));
enddo
```

14) (15 pts) What is the **forward difference** approximation to the 1^{st} derivative of the following function at $x = 3$ using a step size of $h = 0.5$ and a second answer using a step size $h = 0.1$? (No programming here, use the formula by hand just to get an answer.) **Circle** the one you think is most accurate to the actual derivative at 3. (Do the actual calculation using a calculator. Show your equations.)

$$y = x^2 + 2x - 13$$

Actual

$$\frac{d}{dx} = 2x + 2$$

$$\frac{f(3.5) - f(3)}{0.5} = \frac{(3.5^2 + 2(3.5) - 13) - }{0.5}$$

$$\frac{d}{dx}(3) = 2(3) + 2$$

$$= 6 + 2 = 8$$

$$\frac{6.25 - 2}{0.5} = \frac{4.25}{0.5} = 8.5 \leftarrow \text{1st answer}$$

$3.61 + 2(3.1) - 13 = 2.81$

$$\frac{f(3.1) - f(3)}{0.1} = \frac{2.81 - 2}{0.1} = \frac{0.81}{0.1} = 8.1 \leftarrow \text{second answer}$$

closer

Name:_____

15) (5 pts) Write a Fortran do loop that adds the **even** numbers from 2 to n. Just code the do loop. Assume n is already set.

```
do i=2, n                                   sum.
IF (MOD(i, 2) .eq. 0) x = x + n
enddo
```

```
do i = 2, n, 2   ← always just step evenly.
   x = x + i
enddo
```

16) (15 pts) Write a Fortran **recursive** function that adds the **even** numbers from 2 to n. Using this function in some program will look like: **Answer = EvenSum(n)**. Write a complete Fortran function including **all good programming practices.** (Do not write any main program.)

```
recursive function EvenSum(n) result (a)
    implicit none;
    ! variable declaration
    integer :: a, n;
    if (MOD(n,2) .eq. 1) n=n-1; ← This I added to ensure that n is even.
    if (n .eq. 0) return; ! check if n not 0.

    a = n + EvenSum (n-2)  ! add recursive definition
end function EvenSum       ! to sum even numbers.
```