Exam 1 Solutions

CS/ECE 252 Section-2 (MWF 11:00)

Monday, September 28th

Please show any steps you think are necessary to arrive at your answers. Total: 25

1. (a) A computer is defined as a system that is capable of four things. What are these four capabilities?

(2)

arithmetic, storage, branching, I/O

(b) For each of the steps [i (one) through v (five)] in the algorithm written below, select the capabilities needed to execute each step from the list you wrote in part (a). **(3)**

i. Take a number from the user. Place the result in a variable called *n1.*
I/O, storage

ii. Take another number from the user. Place the result in a variable called *n2.*
I/O, storage

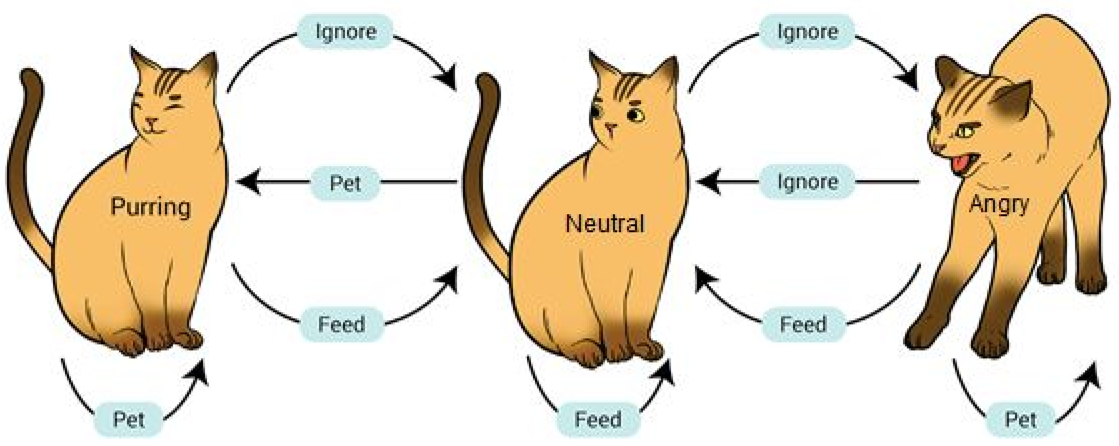
iii. Add *n1* and *n2*. Save the sum in a memory location. We will call this memory location *total*.
arithmetic, storage

iv. Divide *total* by 2. If the remainder is 0, go to step v. Else, go to step vi.
arithmetic, branching

v. Display “total is even” on the monitor. Stop.
I/O

2. Refer to the state diagram below. In order from left to right, the three states are **Purring, Neutral**, and **Angry.** If we begin in the **Neutral** state, what state will the cat be in after each input in the following sequence? For each input, you should indicate the state you move into.

Inputs: Ignore, Ignore, Feed, Ignore, Pet, Pet, Feed, Ignore **(2)**



Action	*Start*	*Ignore*	*Ignore*	*Feed*	*Ignore*	*Pet*	*Pet*	*Feed*	*Ignore*
State	Neutral	Angry	Neutral	Neutral	Angry	Angry	Angry	Neutral	Angry

3. Abstraction is a very powerful tool in programming or developing benefits. List a benefit of abstraction. **(1)**

Various, but 1) user doesn't have to know details to use it 2) system segments can be modified independently 3) specialists can focus on their own area 4) complex systems are easier to break down and plan, etc

4. Circle all properties of algorithmic complexity. **(1)**

a. degree of difficulty of math used in algorithm
b. memory used
c. time taken to write the algorithm
d. time to execute

5. For each **Python** variable name, indicate whether or not it is valid. If it is valid, write "valid". Otherwise, write "invalid". **(4)**

`_num_items`	`if`	`___`	`num_items6`
valid	**invalid**	**valid**	**valid**

`2015sales`	`_2015_sales`	`item 6`	`sixth-item`
invalid	**valid**	**invalid**	**invalid**

6. What is the value stored in x after each of the following sequence of statements are executed? If there is an error in the code, write "no output". **(5)**

```
 i.   x = 5 + 6

ii.   x = "5" + "6"

iii.  a = 10
      2b = 20
      x = a + 2b

iv.   b = 10
      _b = 20
      x = b + _b

 v.   a = 10
      b = 4
      if( (a-b) > 5):
            x = 4
      else:
            x = 6
```

i. 11

ii. "56"

iii. no output

iv. 30

v. 4

7. In the following program, we want to print the 4!, or the factorial of 4.

```
              input  = 4
              answer = 1
              counter = 2

        (i)   while(counter < input):
        (ii)          answer = answer * counter
        (iii)         counter = counter + 1
        (iv)  print(answer)
```

(a) Trace through the code of the while loop using the given variables until the loop terminates. You may not need to use all rows. You should not need to add more rows. If the program does not move onto a next line (i.e. it terminates), write "end" in **next line**. **(3)**

Current line	Calculation performed by this line	Value of `counter`	Value of `input`	Value of `Answer`	Next line:
i	Checking if 2 < 4	2	4	1	ii
ii	Multiplying 1 by 2 and setting answer = 2	2	4	2	iii
iii	Adding 1 to counter	3	4	2	i
i	Checking if 3 < 4	3	4	2	ii
ii	Multiplying 2 by 3 and setting answer = 6	3	4	6	iii
iii	Adding 1 to counter	4	4	6	i
i	Checking if 4 < 4	4	4	6	iv
iv	Printing answer	4	4	6	end

(b) What is the **logical** error in part (a)? **(1)**

It's an "off-by-one" error where the program terminates too quickly because of an incorrect comparison in the while loop. We also accepted a description, such as that it only calculates the factorial of 3!, not 4!

(c) How do you fix it? **(1)**

We can fix it by changing `while(counter < input):` to any of the 3 lines below

```
while(counter <= input):
while(counter - 1 < input):
while(counter < input + 1):
```

Another way to fix it is to, instead of initializing answer to 1, initialize answer to input.

We also accepted changing input = 5, however, that one does only work in this situation.

8. Print the outputs of the following piece of code **(2)**

```
def f(x):
    print(x)
    y = 3
    return y
y = 5
print(f(y))
print(y)
```

5
3
5

9. How many times will the "hello" be printed by the following sequence of statements? What is the value of i at the end? **(Extra credit: 2 pts)**

```
i = 22
while(i > 5):
  print("hello")
  i = i - 3
```

```
6 times
i = 4
```