Topics for CSC 205 Quiz #3 **Wednesday, November 7th**

Quiz #3 will cover Labs 10 & 11, and our class notes and handouts on linked lists, stacks, & queues. Be comfortable with all questions on your linked list worksheet. The following topics are especially important to review

- Definition of a linked list. How do you implement them?
- Be able to write a method to print out the items in a linked list, count the number of items in a linked list, and find the max or min item in a linked list.
- What causes NullPointerException? How is garbage collection of memory handled?
- Be able to write the lines of code to insert an item into a sorted linked list given a previous and current pointer or to delete an item from a sorted linked list.
- The Stack ADT and the operations it supports; Be able to trace through stack code from both the user (client)'s eyes and the implementation eyes; What are the two ways a stack can be implemented? Advantages/disadvantages of both.
- The Queue ADT and the operations it supports; Be able to trace through queue code from both the user (client)'s eyes and the implementation eyes; What are the two ways a queue can be implemented? Advantages/disadvantages of both.

Computer Science 205 Quiz #3

18/50

Friday, November 11th, 2016

50 points

Name : Yu-Ching

1. Consider the following recursive method.

```
private static void mystery (int num)
{
          if (num > 0)
          {
                    System.out.print(num % 2);
                    mystery (num / 2);
          }
}
```

(a) What would be the output of the call mystery(20)? Show a tree diagram with all recursive calls. (6 points)

-6

20 % 2 = 10
mystery (10/2) = 5
5 % 2 = 2, remainder 1
mystery (2/2) = 1
2 % 2 = 0, remainder 1
mystery (0/2) = 0

00101

mystery (20/2) 0
↓
(10/2) 0
↓
(5/2) 1
↓
(2/2) 0 → (1/2) 1

(b) Rewrite the body of this method so that it produces the exact same output using a while loop instead of recursion. (2 points)

-1

```
while (num != 0) {
    System.out.print (num % 2);
    ~~mystery (num/2);~~          num = num / 2
}
```

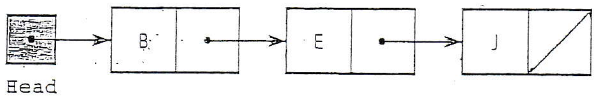
(c) What exactly is the purpose of this mystery method? (2 points)

-2

To see how many calls an integer will call

Print out binary for reverse numbers.

2. Consider the following linked list that is pointed to by a pointer `Head` of type `Node`.



(a) What is the value of each of the following? If an undefined value is printed, indicate so with a `U`. If the expression crashes at run time, indicate so with a `RT Error`. (2 points each)

i. `Head.getNext().getItem()` E

ii. `Head.getNext().getNext()` U

iii. `Head.getNext().getNext().getItem()` J

~~iv.~~ `Head.getNext().getNext().getNext()` U null

-2

v. `Head.getNext().getNext() == Head` False

vi. `Head.getNext().getNext().getNext().getItem()` RT Error

~~(b)~~ Write the line(s) of code that would be needed to add the character `'A'` to the front of this list. Be sure you link your new node properly to the existing list, and reset your `Head`. (4 points)

-4

```
Head = insert(Head, new String("A"));

Head.setItem().top();
```

```
Node prev = new Node("A");
prev.setNext(Head);
Head = prev;
```

~~(c)~~ Suppose your list is as it is originally drawn above, and you decide to delete the character `'E'` (the second item). Write the line(s) of code to perform this task and re-link your list properly. (4 points)

-4

```
Head.getNext().getItem().setItem(null);
```

```
head.setNext(head.getNext().getNext());
```

~~(d)~~ Write the line(s) of code needed to turn this list into a circular linked list. (2 points)

-2

```
Head.getNext().getNext().getNext().setNext(Head)
```

~~3.~~ Write a recursive method named `revPrint` that takes a pointer `Head` that points to the beginning of a linked list, and prints out the contents of that linked list in reverse order. For example, using the linked list in the previous question, your method would print out J E B. (8 points)

−8

```
private static void revPrint (String string) {
  Node head = null;
     head = insert(head, new String ("B"));
     head = insert (head, new String ("E"));
     head = insert (head, new String ("J"));
   if ( head != null) {
      head = head.top();
      return head;
  }
}
```

```
public static void
  revPrint (Node head)
  if (head ! null) {
    sysout(head.getIt
    revPrint (head.getNe
  }
}
```

4. Draw the linked list that would be created by the following program. (6 points)

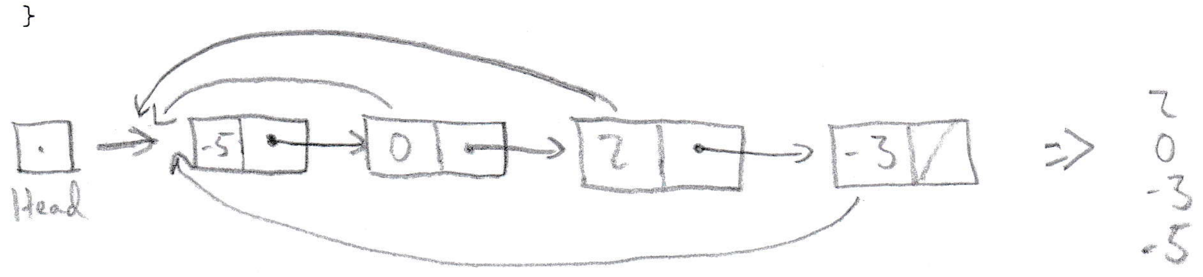
```
public static void main(String args[]) {
   Node head = null;
   head = insert(head, new Integer(-5));
   head = insert(head, new Integer(0));
   head = insert(head, new Integer(2));
   head = insert(head, new Integer(-3));
 }

 private static Node insert(Node head, Comparable newValue) {
   Node prev, curr = head;

   for (prev = null,  curr = head;
             curr != null && newValue.compareTo(curr.getItem()) > 0;
             prev = curr, curr = curr.getNext() ) {}

   Node newNode = new Node(newValue, curr);
   if (prev != null) {
        prev.setNext(newNode);
        return head;
   }
   else
        return newNode;
 }
```

−1



−5   −3   0   2

5. Consider the following class.

```
public abstract class Employee implements Serializable
{
  private String Name;
  private double Wage;

  //  constructor GOES HERE

  public double getWage() {
        return Wage;
  }

  public String getName() {
        return Name;
  }

  public void setName(String Name) {
        this.Name = Name;
  }

  public void setWage(double Wage) {
        this.Wage = Wage;
  }
}
```

(a) Implement a constructor for this abstract class. (2 points)

-2

```
public Employee (String ~~input~~Name, double ~~input~~Wage) {
  this.Name = ~~input~~Name;
  this.Wage = Wage;
```

(b) Explain what other two methods still need to be added. Should either be made abstract? (2 points)

raise method and calculating paychecks method still needs to be added.

✓

The ~~raise~~ method should be abstract
calculate