CS151 Midterm 1

Name:

Start Time:

Finish Time:

I have abided by the Honor Code. I have not discussed this test with anyone.
(Sign below)

If you take this test on separate sheets of paper, make the above your first page.

There are 6 questions in this test. Many of the questions have several parts. Be sure to answer all parts of all questions.

Question	parts	points
1	4	4
2	1	20
3	4	21
4	6	12
5	4	16
6	2	20

Question 1. (4 points, 1 points each) Identify the line(s) of code that best fits the situation described:

A. Declare a variable of a StuffBag that contains only objects of type String
1. StuffBag sb1 = new StuffBag();
2. StuffBag<Object> sb2 = new StuffBag<Object>();
3. StuffBag<String> sb3 = new StuffBag<String>();
4. StuffBag<S> sb4 = new StuffBag<S>();

Ans: 3

B. Declare a variable of a Pair that contains only a String key and an Integer value
1. Pair<String, int> p1 = new Pair<String, int>("A",1);
2. Pair<String, Integer> p1 = new Pair<String, Integer>("A",1);
3. Pair p1 = new Pair("A",1);
4. Pair<Object, Object> p1 = new Pair<Object, Object>("A",1);

ANS: 2

C. Create a class that has two generic parameters
1. public class A<B,B> { … }
2. public class A<C,D> { … }
3. public class A<E><F> { … }
4. public class A[G,H] { … }

ANS: 2

D. Create a class that contains an inner class
1. public class A {
   public class B { … }
   }
2. public class A {
   protected class B { … }
   }
3. public class A {
   }
   private class B{ … }
4. public class A {
   B = new B();
   }

ANS: 2

Question 2 (20 points) : Write a new method for the StuffBag class that returns the contents of the StuffBag in an ArrayList without any duplication.
For your reference, here is the start of the StuffBag class; the only instance variable in the class is stuffArray

```
public class StuffBag<R> implements BagOfStuff<R> {
    /** The array holding the information in the bag */
    private R[] stuffArray;
```

Here is complete top-level documentation of the method you are to write

```
    /**
    * Converts the contents of a Bag into a set.
    * That is, this function returns a new data structure,
    * specifically an ArrayList,
    * that contains every item in the original Bag, exactly once.
    * For instance, if the original bag contained <1,3,1,3,2,1>
    * then the returned ArrayList should contain <1,2,3> in any
    * order.
    * @return An ArrayList that contains the items in the bag,
    *  without duplication
    */
public ArrayList<R> toSet()
```

Other than a constructor, the only methods of ArrayList that you should use are get and add. Since you will be writing this method within the StuffBag class you can use any of its methods.

```
private boolean notIn(R datum, ArrayList<R> aL) {
        for (R inn : aL) {
                if (inn.equals(datum))
                        return false;
        }
        return true;
}
public ArrayList<R> toSet() {
        ArrayList<R> ret = new ArrayList<>();
        for (int i=0; i<stuffArray.length; i++) {
                if (stuffArray[i]!=null)) {
                        if (notIn(stuffArray[i], ret) {
                                ret.add(stuffArray[i]);
                        }
                }
        }
        return ret;
}
```

Question 3: A Pet shelter wants a program in which they can store all the pets in the shelter. So you provide them with ShelterGB from Lecture4. They request that you add one more method with the following top-level documentation.

```
/**
     * Selects a Pet from the shelter for adoption.
     * The pet in that has been in the shelter longest should
     * be selected 50% of the time. The second longest pet
     * should be selected 25% of the time.  The third longest,
     * 12.5% of the time, etc.  It does not matter how much
     * longer one pet has been in the shelter that another, only
     * the order of their arrival.
     * The pet returned from by this function should be removed
     * from the shelter.  If you get to the last animal and have
     * not picked one yet, pick the last animal.
     * @return the Pet to be adopted, or null is there are
     * no pets in the shelter
     */
    public Pet adoptPreferentially()
```

Hint, another way to think about this is to consider the first pet, it has a 50% chance of being selected. If it is not selected, then consider the second pet, it now has a 50% chance of being selected. If that pet is not selected, consider the third pet, it now has a 50% chance of being selected ……

PART A. (3 points) Looking at this request you get the idea that this method would be easier to implement if you changed from a Bag to one of the other Data Structures discussed in class. What data structure would you change to any why.

I would put everything in an ArrayList rather than a Bag. ArrayList preserves order, Bag does not. So, I could guarantee that the oldest pet was in position 0 of the ArrayList

PART B. (2 points) What property of GBShelter would make this change easy.

Encapsulation

PART C. (14 points) Implement the adoptPreferentially method. (there is lots of space on the next page for the implementation)
You will need a random number generator in your method. To get a random number use the following code (which gives you a random number in the range 1..100 inclusive) :

```
Random r = new Random()
int aa = r.nextInt(100)+1;
```

```
// Assume that the ArrayList is named aList
public Pet adoptPreferentially() {
        Random r = new Random()
        for (int i=0; i<aList.size()-1; i++) {
                int aa = r.nextInt(100)+1;
                if (aa<50) {
                        Pet p = aList.get(i);
                        aList.remove(i);
                        return p;
                }
        }
        Pet p = aList.get(aList.size()-1));
        aList.remove(aList.size()-1);
        return p;
}
```

PART D. (2 points) What is the algorithmic complexity of your method. This is for your method, irregardless of its correctness.
Complexity = O(n)

Question 4: Algorithmic Complexity.

PART A: (2 points) Place the following complexity descriptors in order based on their expected asymptotic run-time (ie, when n gets really large).

O(n)
O(lg n)
O(n*n)
O(1)
O(n * lg n)

O(1), O(lg n), O(n), O(n * lg n), O(n*n)

PART B (which consists of 5 parts, 2 points each): For each of the following methods, give its algorithmic complexity. Note that some methods may be used by other methods

Method	Complexity

`public int a1(int[] arr) {`
`  return arr.length;`
`}`
	O(1)

`public int a2(int[] arr) {`
`  int nih = 0;`
`  while (true) {`
`  nih = nih + 10;`
`  if (nih > arr.length * 100)`
`  break;`
`  }`
`  return nih;`
`}`
	O(n)

`public int a3(int[] arr) {`
`  int nih = 0;`
`  for (int i = 0; i < 1000000; i++) {`
`  while (true) {`
`  nih = nih + 10;`
`  if (nih > arr.length * 100)`
`  break;`
`  }`
`  }`
`  return nih;`
`}`
	O(n)

Method	Complexity

`public double a4(int[] arr) {`
`  double nih = 0;`
`  for (int i = 0; i < arr.length; i++) {`
`  nih++;`
`  }`
`  while (true) {`
`  nih = nih * 1.5;`
`  if (nih > arr.length * 100)`
`  break;`
`  }`
`  return nih;`
`  }`
	O(n)
	
	There are two loops, the first is O(n), the second appears to be O(lg n) — actually it is O(1). when loops appear sequentially, you add . So O(n)+O(1) == O(n)

`public int a5(int[] arr) {`
`  int ii = 0;`
`  while (ii < arr.length) {`
`  ii = ii + 1 + a2(arr);`
`  }`
`  return ii;`
`  }`
	O(n*n)

Question 5: (4 parts, 4 points each) For each of the programs in the left column, the output is given in the right column. Explain, briefly, why the output is what it is. Line numbers are provided for easy reference. The first program is an example; it is actually pretty long for what the program does.

Sample Program	**Output**

1 public class Outp {
2 public static void main(String[] args) {
3 System.out.println("main");
4 }
5 }
	main

Sample Explanation:
When main method executes, the println in line 3 executes, which prints main

Q5 Part A: Program	**Output**

1 public class Out1 {
2 public static void main(String[] args) {
3 new Out1().work();
4 }
5
6 private class Out1b extends Out1 {
7 public String toString() {
8 return "B" + super.toString();
9 }
10 }
11
12 public void work() {
13 Out1b ob = new Out1b();
14 System.out.println(ob + ob.toString());
15 }
16 public String toString() {
17 return “aa";
18 }
19 }
	BaaBaa
	
	Call to toString results in call to B + superToString which gives aa
	
	in work func call println on ob which implicitly calls tostring, then explicitly calls tostring

Q5 Part B: Program	Output

```
 1  public class Out2 {
 2      public Out2() {
 3          System.out.println("Out2a");
 4      }
 5
 6      public Out2(int iii) {
 7          this();
 8          for (int i = 0; i < iii; i++)
 9              System.out.print("Out2");
10          System.out.println();
11      }
12
13      public String a(int iii) {
14          System.out.println(iii);
15          return "a1";
16      }
17
18      public String a(int iii, int jjj) {
19          System.out.println((iii - jjj));
20          return "a2";
21      }
22      public static void main(String[] args) {
23          Out2 o2a = new Out2();
24          Out2 o2b = new Out2(3);
25          System.out.println(o2a.a(4) + o2b.a(4, 1));
26      }
27  }
```

```
Out2a
Out2a
Out2Out2Out2
4
3
a1a2
```

call to Out2 constructor prints out2a
then call to Out2 1 ar constructor prints Out2a due to this(), followed by Out2 3 times on same line.
The call to one arg a(4) prints a1 followed by a2 from the 2 args a method. Before that the 2 ares a method prints its two args

Q5 Part C: Program	Output

1 public class Out3 {
2 public void ss(String strin) {
3 char[] carray = new char[strin.length()];
4 for (int i = 0; i < strin.length(); i++) {
5 carray[i] = strin.charAt(i);
6 }
7 for (int i = 0; i < carray.length / 3; i++) {
8 char tm = carray[i];
9 carray[i] = carray[carray.length - 1 - i];
10 carray[carray.length - 1 - i] = tm;
11 }
12 for (int i = 0; i < carray.length; i++) {
13 System.out.print(carray[i]);
14 }
15 System.out.println();
16 }
17
18 public static void main(String[] args) {
19 new Out3().ss("thisisatest");
20 }
21 }
	tsesisatiht
	
	The fun ss swaps the beginning and end characters for the first third of the provided string.
	
	Then prints the result.

Q5 Part D: Program	Output

1 import java.util.ArrayList;
2
3 public class Out4 {
4 private ArrayList<Integer> aa = new ArrayList<>();
5
6 public Out4(int num) {
7 aa.add(0);
8 for (int ii = 1; ii < num; ii = ii * 2) {
9 aa.add(aa.get(aa.size()-1) + ii);
10 }
11 }
12
13 public String toString() {
14 if (aa.size() > 10)
15 return "" + aa.get(aa.size()/4);
16 else
17 return "-1";
18 }
19
20 public static void main(String[] args) {
21 System.out.println(new Out4(1));
22 System.out.println(new Out4(100));
23 System.out.println(new Out4(1000));
24 System.out.println(new Out4(100000));
25 }
26 }
	−1
	−1
	3
	15
	
	The first 4 this filled
	into the array list are
	1
	3
	7
	15
	That is enough to see
	that the output is as it
	is because you only have
	to know that much of any
	of the arrays

Question 6: (20 points) Use double hashing to fill a hashtable of size 11.
Keys are composed of Strings of length 3. All strings are composed only of the digits between 1 and 9, inclusive.
Values are single characters.

h(x) = (digit1*9+digit2*3+digit3) % (size of hashtable)
h2(x) = Integer.parseInt(x)%5 +1

For example if the key is 123, then
h(123) = (1*9+ 2*3 + 3)%11 = (9+6+3)%11 = 18%11 = 7
h2(123) = 123 % 5 + 1 = 3+1 = 4

Now consider the following additions

Add <“111”, a>

Add <“213”, b>

Add <“314”, c>

Add <“431”, d>

Add <“463”, e>

Add <“711”, f>

PART A: (12 points)Show the contents of the hash table after the 6 additions above.

PART B: (8 points) The hash table is now kind of full. So grow the hashtable to a size of 20 (20 is a horrible size for a hash table, but it makes the math a lot easier) and adjust h(x) appropriately. Move all of the entries into the new hashtable. Show the new hashtable after everything is moved in.

		h(x)	h2(x)	Location in h size 11	h(x) for size 20	Location in ht size 20
<111,a>	**13**	2	2	(2) 2	13	13
<213,b>	**24**	2	4	(2+4) 6	4	4
<314,c>	**34**	1	5	(1) 1	14	14
<431,d>	**46**	2	2	(2+2) 4	6	6
<463,e>	**57**	2	4	(2+4+4) 10	17	17
<711,f>	**67**	1	1	(1+1+1) 3	7	7

(page intentionally left blank)

	ht size 11	ht size 20
0		
1	314,c	
2	111,a	
3	711,f	
4	431, d	213,b
5		
6	213,b	431,d
7		711,f
8		
9		
10	463,e	
11		
12		
13		111,a
14		314,c
15		
16		
17		463,e
18		
19		