

CS102, FINAL EXAM, MONDAY, DECEMBER 21, 2009,
PROF. LOFTIN

SOLUTIONS

- (1) (6 pts) Let f_k be defined by

$$f_0 = 3, \quad f_k = 1 + 2f_{k-1} \quad \text{for } k \geq 1.$$

Write a recursive method

```
public static int f(int k)
```

whose return value is f_k .

```
public static int f(int k){
    if (k==0)
        return 3;
    else
        return 1 + 2*f(k-1);
}
```

- (2) (5 pts) Assume that LStack is an implementation of the Stack interface containing a stack of integers. What is the output to the screen of the following program?

```
public class StackProblem{
    public static void main(String[] args){
        Stack s = new LStack();
        s.push(5);
        s.push(7);
        s.push(s.pop() + s.pop());
        s.push(10);
        s.push(15);
        System.out.println(s.pop());
        int x = s.pop(), y = s.pop();
        System.out.println(y-x);
    }
}
```

Solution:

15

2

- (3) (15 pts) Consider the following method

```
public static int p(int n){
    if (n==0)
        return 1;
    else
        return 3*p(n-1);
}
```

- (a) What is the return value $p(4)$?

Solution: 81

- (b) Give an explicit mathematical non-recursive formula for the return value $p(n)$ for n a positive integer.

Solution: 3^n

- (c) What happens if -1 is passed into the method p as the parameter n ?

Solution: There is an infinite recursion. Eventually, the computer will run out of available memory for the stack space required for each method call.

- (d) Rewrite the method p so that it avoids the problem in part (c). In other words, it should behave well for every integer value of the parameter n .

Solution:

```
public static int p(int n){
    if (n<=0)
        return 1;
    else
        return 3*p(n-1);
}
```

- (4) (6 pts) What is the output of the following program?

```
public class Problem{
    public static void main (String[] args){
        int[] a = {3,4,7,10,2,1};
        int[] b = {3,6,9,8,0};
        printOut(a);
        changeB(b);
        printOut(b);
        changeA(a);
        printOut(a);
    }
    public static void changeA (int[] ar){
        ar = new int[10];
        for (int i=0; i<10; i++)
            ar[i] = i*i;
    }
}
```

```

    }
    public static void changeB (int[] ar){
        for (int i=0; i<ar.length; i++)
            ar[i] += i;
    }
    public static void printOut (int[] ar){
        for (int i=0; i<ar.length; i++)
            System.out.print(ar[i] + " ");
        System.out.println();
    }
}

```

Solution:

```

3 4 7 10 2 1
3 7 11 11 4
3 4 7 10 2 1

```

- (5) (5 pts) What is the return value $r(43)$? Justify your answer.

```

public static String r (int n){
    if (n<5)
        return n + "";
    else
        return r(n/5) + n%5 + "";
}

```

Solution: "133". This is because of the following computation:

```

r(43) = r(8) + 3 + ""
       = r(1) + 3 + "" + 3 + ""
       = 1 + "" + 3 + "" + 3 + ""
       = "133"

```

Note that the return value $r(n)$ is the base-5 representation of the positive integer n .

- (6) (6 pts) Trace through the insertion sort on the array of integers

```

3 7 0 2 13 -1 6

```

In particular, write out the elements of the array after each pass of the algorithm.

Solution:

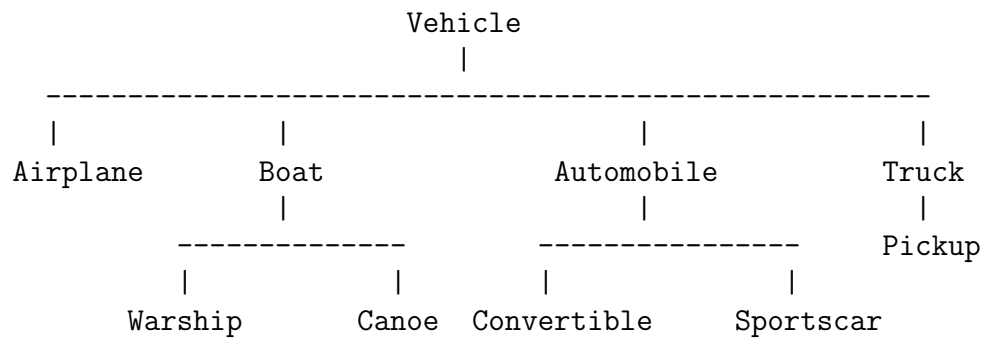
```

3 7 0 2 13 -1 6
3 7 0 2 13 -1 6
0 3 7 2 13 -1 6
0 2 3 7 13 -1 6
0 2 3 7 13 -1 6
-1 0 2 3 7 13 6

```

-1 0 2 3 6 7 13

- (7) (10 pts) Write a class hierarchy containing the following classes: Airplane, Vehicle, Boat, Canoe, Automobile, Truck, Convertible, Warship, Pickup, Sportscar.

Solution:

- (8) (6 pts) What is the output to the screen?

```

public class Driver{
    public static void main (String[] args){
        Speak sp = new Speak();
        sp.message();
        sp = new SpeakChild();
        sp.message();
    }
}
public class Speak{
    public void message(){
        System.out.println("Speak only when spoken to.");
    }
}
public class SpeakChild extends Speak{
    public void message(){
        System.out.println("Children should be seen and not heard.");
        super.message();
    }
}
  
```

Solution:

```

Speak only when spoken to.
Children should be seen and not heard.
Speak only when spoken to.
  
```

- (9) (8 pts) Write a method

```

public static Comparable max (Comparable[] ar)
  
```

which returns the maximum value of an array `ar` of objects which implement the `Comparable` interface.

Solution:

```
public static Comparable max (Comparable[] ar){
    if (ar.length == 0)
        return null;
    Comparable ret = ar[0];
    for (int i=1; i<ar.length; i++)
        if (ret.compareTo(ar[i]) < 0)
            ret = ar[i];
    return ret;
}
```

(10) (18 pts) Consider the interface

```
public interface Shape{
    public double area();
    public double perimeter();
}
```

(a) Write a class

`public class Rectangle` implements `Shape` whose constructor stores the height and width of the rectangle. Implement the `Shape` interface. Recall the area of a rectangle is hw for h the height and w the width, while the perimeter is $2h + 2w$.

Solution:

```
public class Rectangle implements Shape{
    private double height, width;
    public Rectangle(double h, double w){
        height = h;
        width = w;
    }
    public double area(){
        return height*width;
    }
    public double perimeter(){
        return 2*height + 2*width;
    }
}
```

(b) Write a class

```
public class Square extends Rectangle
```

whose constructor stores the side-length of the square. Use inheritance to ensure the area and perimeter are correctly calculated.

Solution:

```
public class Square extends Rectangle{
    public Square (double s){
        super(s,s);
    }
}
```

(c) Write a class

`public class Triangle implements Shape` whose constructor stores the three sides of a triangle a, b, c . The perimeter of a triangle is given by $a + b + c$, while the area is given by the formula

$$\sqrt{s(s-a)(s-b)(s-c)},$$

where $s = (a + b + c)/2$. Recall the `Math.sqrt` method computes the square root in Java.

Solution:

```
public class Triangle implements Shape{
    private double a, b, c;
    public Triangle (double aa, double bb, double cc){
        a = aa;
        b = bb;
        c = cc;
    }
    public double perimeter(){
        return a+b+c;
    }
    public double area(){
        double s = (a+b+c)/2;
        return Math.sqrt(s * (s-a) * (s-b) * (s-c));
    }
}
```

(11) (5 pts) Let `q` represent an initially empty queue of integers. List, in order, the elements of `q` after the following code executes. Justify your answer.

```
q.enqueue(5);
q.enqueue(10);
for (int i=0; i<2; i++){
    int x = q.dequeue();
    if (x%2 == 0)
```

```

        q.enqueue(x);
    else{
        q.enqueue(i);
        q.enqueue(x-1);
    }
}

```

Solution: 0 4 10

- (12) (8 pts) A string is said to be a palindrome if it is the same string when all the characters are written in the reverse order. For example, “a”, “eve”, “star rats”, and “toot” are palindromes, while “evaluate” is not. Write a recursive method

```
public static boolean palindrome(String s)
```

which returns true if *s* is a palindrome and false otherwise.

Recall some methods of the `String` class:

```

char charAt (int index) // character at index
int length() // length of the string
String substring (int offset, int endIndex)
    // returns substring from index offset to endIndex-1

```

Hint: Test whether the first and last characters of *s* are equal, and if so, recursively call the method with a smaller substring. What should the base case be?

Solution:

```

public static boolean palindrome (String s){
    int len = s.length();
    if (len < 2)
        return true;
    else if (s.charAt(0) == s.charAt(len-1))
        return palindrome(s.substring(1,len-1));
    else
        return false;
}

```

- (13) (20 pts) Consider the class `MyList` (as discussed in class):

```

public class MyList{
    private MyListNode head;
    public MyList(){ head = null; }
    public void addAtHead (int value){
        MyListNode temp = head;
        head = new MyListNode(value);
        head.next = temp;
    }
}

```

```

public void add (int index, int value){
    if (index==0)
        addAtHead(value);
    else{
        MyListNode prev = head, curr = head.next;
        for (int i=1; i<index; i++){
            prev = curr;
            curr = curr.next;
        }
        MyListNode newNode = new MyListNode(value);
        prev.next = newNode;
        newNode.next = curr;
    }
}
public String toString(){ // fill in
}
private class MyListNode{
    public int num;
    public MyListNode next;
    public MyListNode(int n){
        num = n;
        next = null;
    }
}
}

```

- (a) Write the `toString` method of the `MyList` class (given on the previous page), whose return value should return all the elements of the list, in order, separated by spaces.

Solution:

```

public String toString(){
    String ret = "[ ";
    MyListNode curr = head;
    while (curr != null){
        ret += curr.num + " ";
        curr = curr.next;
    }
    return ret + "]";
}

```

- (b) What is the output of the following driver class?

```

public class MyListDriver{
    public static void main (String[] args){

```

```

        MyList a = new MyList();
        a.addAtHead(5);
        a.addAtHead(3);
        System.out.println(a);
        a.add(2,4);
        a.add(1,6);
        System.out.println(a);
    }
}

```

Solution:

```

[ 3 5 ]
[ 3 6 5 4 ]

```

- (c) Write a class `ListIndexException` which extends the `Exception` class, and sets an appropriate exception message.

Rewrite the `add` method of the `MyList` class above so that a `ListIndexException` is thrown if it is attempted to add a new element to the list at an inappropriate index (so, for example, if the list `b` contains `5 9`, then the method call `add` will throw the exception if the `index` parameter is less than 0 or more than 2.)

Solution:

```

public class ListIndexException extends Exception{
    public ListIndexException(){
        super("List Index Out of Bounds");
    }
}

public void add (int index, int value) throws ListIndexException{
    if (index<0)
        throw new ListIndexException();
    if (index==0)
        addAtHead(value);
    else{
        if (head == null)
            throw new ListIndexException();
        MyListNode prev = head, curr = head.next;
        for (int i=1; i<index; i++){
            prev = curr;
            if (prev == null)
                throw new ListIndexException();
            curr = curr.next;
        }
    }
}

```

```

    }
    MyListNode newNode = new MyListNode(value);
    prev.next = newNode;
    newNode.next = curr;
  }
}

```

- (14) (5 pts) What is the output of the following code fragment?

```

int[] ar = {3,7,2,4};
for (int i=0; i<ar.length; i++){
  for (int j=0; j<ar[i]; j++)
    System.out.print("*");
  System.out.println("oo");
}

```

Solution:

```

***oo
*****oo
**oo
****oo

```

- (15) (8 pts) Write a method `concatenateArrays` which takes as parameters two arrays of `doubles`, and returns a new array which consists of the elements of the first array followed by the elements of the second array.

Solution:

```

public static double[] concatenateArrays (double[] a, double[] b){
  double[] ret = new double[a.length + b.length];
  for (int i=0; i<a.length; i++)
    ret[i] = a[i];
  for (int j=0; j<b.length; j++)
    ret[a.length + j] = b[j];
  return ret;
}

```

- (16) (6 pts) What is the output to the screen?

```

public class Main{
  public static void main(String[] args){
    f(2);
  }
  public static void f(int n){
    if (n>0){
      System.out.println("beginning of f: n = " + n);
      g(n);
    }
  }
}

```

```
        System.out.println("end of f: n = " + n);
    }
}
public static void g(int n){
    if (n>0){
        System.out.println("beginning of g: n = " + n);
        f(n-1);
        System.out.println("end of g: g = " + n);
    }
}
}
```

Solution:

```
beginning of f: n = 2
beginning of g: n = 2
beginning of f: n = 1
beginning of g: n = 1
end of g: n = 1
end of f: n = 1
end of g: n = 2
end of f: n = 2
```