Unit 2 – ICS4U0 – Methods and Recursion

Sample Test – February 28, 2025

Name: Gorski

Total	%	Knowledge	Communication	Thinking	Application
(96)	%	(18)	(28)	(22)	(28)

Knowledge

1. Trace what is happening in the following array, using the memory diagram/tracing chart provided. /8

***Riddle:* What tastes better than it smells?**

Answer key: 0=a, 1=p, 2=g, 3=n, 4=u, 5=t, 6=o, 7=s, 8=e, 9=u, 10=h, 11=m, 12=i

```
int s = 1;
int a [] = {0, 7, 2, 5, 5, 2};
           a [3] = 2;
           a [5] = 8;
           a [s] --;
           a [0] = a [4];
           a[4] = s + 3;
           a[2] ++;
```

[0]	[1]	[2]	[3]	[4]	[5]
0	7	2	5	5	2
0	7	2	(2)	5	2
0	7	2	2	5	(8)
0	(6)	2	2	5	8
(5)	6	2	2	5	8
5	6	2	2	(4)	8
5	6	(3)	2	4	8
t	o	n	g	u	e

Riddle Answer: tongue

2. Using the method signatures, call the methods. /10

(a) Signature: `public int jet(double price)`

anything, but a double

```
double price = 1.412;
int ans = jet(price);
```

(b) Signature: `public double glider()`

```
double ans = glider();
```

(c) Signature: `public String drone(char pilot)`

anything, but a char single quotes on a char

```
char pilot = 'y';
String ans = drone(pilot);
```

(d) Signature: `public void helicopter(int controls)`

anything, but an int

```
int controls = 5;
helicopter(controls);
```

Communication

3. What does ORATE stand for? /5

Organization	Reusability	Abstraction	Testing	Extensibility

4. Consider this recursive method, identify the following: /9

```
public int catapult (int n) {
  if (n == 1)
    return 2;
  else
    return catapult (n - 1) * n;
}
```

Base case

Recursive case

(a)	The return type	int
(b)	The method name	catapult
(c)	The parameter name	n
(d)	The parameter type	int
(e)	Circle **and** label the base case and recursive case	☐

Evaluate the following:

(f) `catapult(1);` 2

(g) `catapult(2);` 4

(h) `catapult(0);` stack overflow error

because base case is == instead of >=

5. Provide the phrase or number required. /10

120	(a)	The answer to the 5! (five factorial)
Recursive	(b)	A method that calls itself.
loop	(c)	Every recursive method can be re-coded as this.
1 1 2 3 5 8	(d)	The first 6 terms of the Fibonacci sequence.
probability	(e)	What are factorials used for in math?
extensibility	(f)	By putting code in one place, changes are easier. (ORATE)
proportion of shell coils	(g)	Where does the Fibonacci sequence appear in nature? many others. use the word "proportion of" in front.
Return type	(h)	The name of the output of a method.
Stack Overflow	(i)	The equivalent of an infinite loop in recursion.
Base Case	(j)	The equivalent of a loop stopping condition in recursion.

6. (a) Which of the ORATE principles applies best to the Pizza Party Program? (2 points, one sentence each) /2

specifically refer to it

Reusability means that, once written, a method can be called multiples times to avoid repeating code. In the Pizza Party program, two methods, rollBig and rollSmall, were needed to change all the JLabels and JButtons. This avoided duplicated ifs in ActionPerformed and avoided copy/paste errors.

specifics here

(b) Why are parameters critical to recursive methods? (2 points, one sentence each) /2

Parameters are used in both the base case and the recursive case to control the repetition. In the base case, the parameter is used to stop the repetition and start finding answers. In the recursive case, parameters get smaller and smaller in the method calls, progressing to the base case.

Thinking

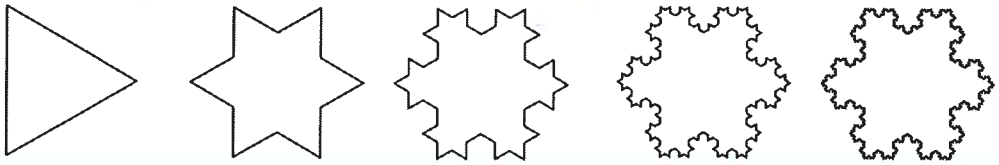
7. Trace the following method with the method call `System.out.println("" + wing(5));` /7

```
public int wing (int n) {
    if (n <= 1)
        return 2;
    else
        return wing (n - 1) + wing (n - 3);
}
```

one back three back

wing(-1)	wing(0)	wing(1)	wing(2)	wing(3)	wing(4)	wing(5)
2	2	2	= wing(1) + wing(-1) = 2 + 2 = 4	= wing(2) + wing(0) = 2 + 4 = 6	= wing(3) + wing(1) = 6 + 2 = 8	= wing(4) + wing(2) = 8 + 4 = 12

8. In a Koch snowflake (which is a recursive shape), each new side of the shape is obtained from the previous one by deleting the middle third of every segment and replacing it with two sides of an equilateral triangle.



(a) If the side of the initial triangle is 81 cm, what is the perimeter of the first 4 generations? /4

Generation	0	1	2	3	4
Number of Line Segments	3	12	48	192	768
Length of 1 Line Segment	81	27	9	3	1
Total Perimeter	243	324	432	576	768

×4
÷3

(b) Write a recursive method to find the number of line segments after n generations. /6

```
public   int   line   ( int   n ) {
method start word   return type (output)   method name (whatever)   parameter type   parameter name (input)

    if( n <= 0 )
        return 3;
    else
        return line(n-1) * 4;
}
```

must match

match 3 times

interestingly, the Koch snowflake has a finite area and an infinite perimeter as n approaches ∞. It can be carpeted, but not fenced.

9. Circle and correct 5 errors. The code is used to switch the turn in a dice game. /5

```
public void
private double endTurn (int n) {
  if (turn == 'x') {
     score1 += dice [n];
     turn = 'o';
     turnPic.setIcon (createImageIcon ("oturn.png"));
  } ELSE {   else
     score2 += dice [n];
     turn = '0';   X
     turnPic.setIcon (createImageIcon ("xturn.png"));
  }
  score.setText ("X's score: " + score1 + ", O's score: " + score0);   2
}
```

Application

JLabel names here

10. Create a method to change the "Add Them" images. Then, call it in the actionPerformed method. /7

(a) Fill in the setPic method: (1) (2)

```
public void setpic(JLabel b, int num) {

   if (num == 1 )
      b.setIcon(createImageIcon( one.png ));
   else if (num == 2 )
      b.setIcon(createImageIcon( two.png ));
   else if (num == 3 )
      b.setIcon(createImageIcon( three.png ));
   else
      b.setIcon(createImageIcon( four.png ));
}
```

The picture names, the corresponding random numbers.

one.png, 1

two.png, 2

three.png, 3

four.png, 4

(b) Fill in the actionPerformed method, calling the setPic method for each player.

```
public void actionPerformed(ActionEvent e) {
//random number for each player
  int r1 = (int) (Math.random()*4)+1;
  int r2 = (int) (Math.random()*4)+1;
  int r3 = (int) (Math.random()*4)+1;
  int r4 = (int) (Math.random()*4)+1;
 //the method calls:
   setPic ( p1 , r1 );
   setPic ( p2 , r2 );
   setPic ( p3 , r3 );
   setPic ( p4 , r4 );
}
```

int names here.

① JLabel ② int — make order match parameters of method.

11. Write a recursive method for the following sequence: 10 15 20 25 30 35 40 45 50 55 … /6

For example, seq(1) returns 10 and seq(3) returns 20.

```
public  int  seq ( int  n ) {
 method start word | return type (output) | method name (whatever) | parameter type | parameter name (input)

   if( n <= 1 )
      return 10 ;
   else
      return seq(n-1) + 5 ;
}
```

match. match.

don't forget to call method to get previous term.

12. Consider this constructor from a dice game called Chicago. In this game, two players take turns rolling a pair of dice to get a target sum of 2, then 3, then 4, all the way to 12. Once a player has rolled their current target, the target increases to the next sum. The first player to get all the sums, in order, wins.

```
System.out.println ("Round " + round + " ************");
while (tar [0] != 13 && tar [1] != 13){
  if (turn == 1) {
    System.out.println ("Player 1, your target sum is "+tar[0]);
    int d1 = (int) (Math.random () * 6) + 1;
    int d2 = (int) (Math.random () * 6) + 1;
    int sum = d1 + d2;
    System.out.println ("Roll: " + d1 + "+" + d2 + "=" + sum);
    if (tar [0] == sum){
       System.out.println ("You made your target!");
       tar [0]++;
       System.out.println ("Next, target will be " + tar [0]);
    } else {
       System.out.println ("You didn't make your target.");
       System.out.println ("Next, target is still " + tar [0]);
    }
    if (turn == 1) {
       turn = 2;
    } else {
       turn = 1;
       round++;
       System.out.println ("Round " + round + " **********");
    }
  }

  else if (turn == 2) {
    System.out.println ("Player 2, your target sum is "+tar[1]);
    int d1 = (int) (Math.random () * 6) + 1;
    int d2 = (int) (Math.random () * 6) + 1;
    int sum = d1 + d2;
    System.out.println ("Roll: " + d1 + "+" + d2 + "=" + sum);
    if (tar [1] == sum) {
      System.out.println ("You made your target!");
      tar [1]++;
      System.out.println ("Next, target will be " + tar [1]);
    } else {
       System.out.println ("You didn't make your target.");
       System.out.println ("Next, target is still " + tar [1]);
    }
    if (turn == 1){
       turn = 2;
    } else {
       turn = 1;
       round++;
       System.out.println ("\nRound " + round + " ***********");
       }
    }
}
```

*make them more of a square

to be clear

(a) ❑ **Circle 2 sets of repeated code in the previous section.** /2

(b) Write the takeTurn method using the parameters given. /6

```
public void takeTurn (String name, int n) {
  System.out.println (name + ", your target sum is " + tar[n]);
  int d1 = (int) (Math.random () * 6) + 1;
  int d2 = (int) (Math.random () * 6) + 1;
  int sum = d1 + d2;
  System.out.println ("Roll " + d1 +"+"+d2 +"=" + sum);
  if (tar [n] == sum){
     System.out.println ("You made your target!");
     tar [n]++;
     System.out.println ("Next, target will be "+tar [n]);
  } else {
     System.out.println ("You didn't make your target.");
     System.out.println ("Next, target is still "+tar [n]);
  }
  switchTurn();
}
```

(c) Write the switchTurn method. It is called in takeTurn (above in part b) /3

```
public void switchTurn (){
   if (turn == 1){
      turn = 2;
   } else {
      turn = 1;
      round ++;
      System.out.println ("\nRound "+ round +" *********");
   }
}
```

(d) Rewrite the constructor to call the takeTurn method (coded in part b). /4

```
System.out.println ("Round " + round + " ************");
while (tar [0] != 13 && tar [1] != 13){
   if (turn == 1)
      takeTurn ("Player 1", 0);
   else if (turn == 2)
      takeTurn ("Player 2", 1);
}
```

name

int

see parameters in b for order.