Unit 2 – ICS4U0 – Methods and Recursion

Sample Test – Thursday February 26, 2026

Name: Gorski

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

Knowledge

1. Trace the code, using the memory diagram/tracing chart provided. /8

***Riddle:* What has a neck but no head?**

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

```
int s = 2;
int a [] = {8, 6, 4, 2, 1, 0};
a [5] = 1;
a [4] = 8;
a [3] --;
a [2] = a [3];
a [4] = a [5] + 3;
a [0] ++;
```

[0]	[1]	[2]	[3]	[4]	[5]
8	6	4	2	1	0
8	6	4	2	1	(1)
8	6	4	2	(8)	1
8	6	(3)	2	8	1
8	6	(2)	2	8	1
8	6	2	2	(5)	1
(9)	6	2	2	5	1
b	o	t	t	l	e

Riddle Answer: ....battle.....

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

(a) Signature: `public void maple (double height)`

Call:

double height = IO.inputDouble("How tall? ");

maple(height);

(b) Signature: `public double danger(int alligator)`

Call:

int alligator = IO.inputInt("How many alligators? ");

double ans = danger(alligator);

(c) Signature: `public void okapi()`

Call:

okapi();

(d) Signature: `public int turkeySize(String food)`

Call:

String food = IO.inputString("What type of food? ");

int answer = turkeySize(food);

Communication

3. What does ORATE stand for? /5

Organization	Reusability	Abstraction	Testing	Extensibility

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

```
public int mandlebrot (int n) {
  if (n <= 2)
    return 10;
  else
    return mandlebrot (n - 2) + 20;
}
```

Base Case

Recursive Case

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

0 1 2 3 4 5
Evaluate the following: 10 10 10 30 30 50

(f) `mandlebrot(1).` 10

(g) `mandlebrot(4).` 30

(h) `mandlebrot(0).` 10

6. Provide the phrase or number required. /10

1 2 3 4 5 6
1 1 2 3 5 8

8	(a)	The 6th Fibonacci number. (The 1st is 1 and the 2nd is also 1)
Rabbit Populations	(b)	Where does the Fibonacci sequence appear in nature? face proportions, seashell turn proportions, flower petals
24	(c)	The answer to 4! (four factorial). 1x2x3x4
Probability Calcs.	(d)	What are factorials used for in math?
Testing	(e)	White and black box verification have fewer cases. (ORATE)
Extensibility	(f)	By putting code in one place, changes are easier. (ORATE)
Abstraction	(g)	To use methods, understanding is not needed. (ORATE)
Parameter	(h)	The name of the input to a method.
Stack Overflow Error	(i)	The equivalent of an infinite loop in recursion.
Base Case	(j)	The equivalent of a loop stopping condition in recursion.

7. Which of the ORATE principles applied best to the **Pizza Party** program? (2 points, one sentence each) /2

fully discuss with specifics

Reusability means that, once written, a method can be called multiple times. In the Pizza Party program, two methods (rollBig & rollSmall) were called to change all the JLabels & JButtons. This avoided duplicated ifs in ActionPerformed, avoiding copy/paste errors.

8. Why are if statements important to recursion? (2 points, one sentence each) /2

If statements are used to distinguish the base case from the recursive case; this allows the code to CHOOSE the correct piece as needed.

The Base Case is selected for small values of the parameter to stop while the recursive case is selected to build on previous generations.

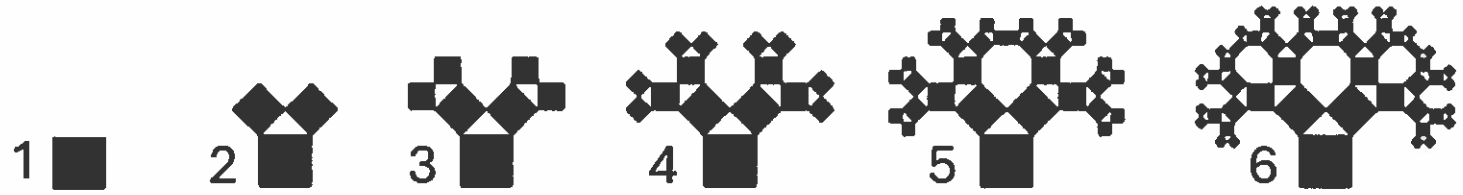
Thinking

9. Fill in the chart to trace the following method call `System.out.println(avatar(5));` /7

```
public int avatar (int n) {
    if (n < 2)
        return 5;
    else
        return n + avatar (n - 2);
}
```

avatar(0)	avatar(1)	avatar(2)	avatar(3)	avatar(4)	avatar(5)
= 5	= 5	= 5 + 2 = 7	= 5 + 3 = 8	= 7 + 4 = 11	= 8 + 5 = 13

8. In a right-angle fractal (which is a recursive shape), each level of the tree is obtained by drawing smaller squares as the end of the previous square. In the space between the old and new squares, a right angle is formed.



(a) How many branches exist in each generation of the tree fractal? /4

Generation	1	2	3	4	5	6	7	8
New squares	1	2	4	8	16	32	64	128
Total squares	1	3	7	15	31	63	127	255

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

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

```
if( n <= 1 )
    return 1;
else
    return squares (n-1) * 2 + 1;
}
```

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

```
public void endTurn (None n) {
  if (turn == 'x') {
     score1 += dice [n];
     turn = 'o';
     turnPic.setIcon (createImageIcon ("oturn.png"));
  } else {
     score2 += dice [n];
     turn = 'x';
     turnPic.setIcon (createImageIcon ("xturn.png"));
  }
  score.setText ("X's score: " + score1 + ", O's score: " + score2);
}
```

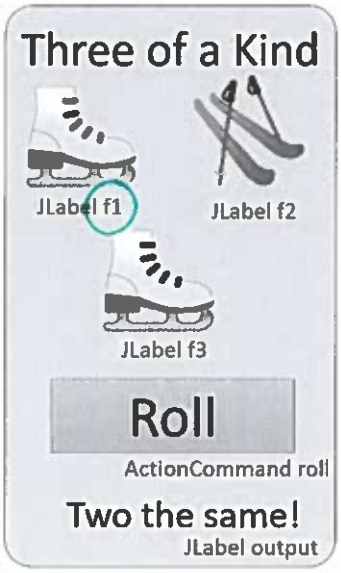
Remember to cross out to remove if needed.

Application

10. Fill in the method to change images in the "Three of a Kind" app. Then, call it in ActionPerformed. /7

(a) Fill in the setPic method:

```
public void setpic(JLabel t, int num) {
    if (num == 0 )
       t.setIcon(createImageIcon("ski.png"));
    else if (num == 1 )
       t.setIcon(createImageIcon("scarf.png"));
    else if (num == 2 )
       t.setIcon(createImageIcon("skate.png"));
    else
       t.setIcon(createImageIcon("gloves.png"));
}
```



The picture names, the corresponding random numbers.


ski.png, 0


scarf.png, 1


skate.png, 2


gloves.png, 3

(b) Call the setPic method for each picture in actionPerformed.

```
public void actionPerformed(ActionEvent e) {

//random number for each
  int r1 = (int) Math.random()*4;
  int r2 = (int) Math.random()*4;
  int r3 = (int) Math.random()*4;

//three method calls:
  setPic ( f1 , r1 );
  setPic ( f2 , r2 );
  setPic ( f3 , r3 );
}
```

11. Write a recursive method for the following sequence: 9 12 15 18 21 24 27 30 33 36 ... /6
For example, seq(1) returns 9 and seq(3) returns 15.

```
public int seq ( int n ) {
   if( n <= 1 )
      return 9;
   else
      return seq (n-1) + 3;
}
```

method start word; return type (output); method name (see question); parameter type; parameter name (input) — must match

must match

12. Consider this win method from a tic tac toe game.

```
public void win () {
    if (b [0] [0] != 'b' && b [0] [0] == b [0] [1] && b [0] [1] == b [0] [2]) {
        String w = b [0] [0] + " has won!!";
        JOptionPane.showMessageDialog (null, w, "Horizontal Win", JOptionPane.ERROR_MESSAGE);
        button [0] [0].setEnabled (false);
        button [0] [1].setEnabled (false);
        button [0] [2].setEnabled (false);
    }
    if (b [1] [0] != 'b' && b [1] [0] == b [1] [1] && b [1] [1] == b [1] [2]){
        String w = b [1] [0] + " has won!!";
        JOptionPane.showMessageDialog (null, w, "Horizontal Win", JOptionPane.ERROR_MESSAGE);
        button [1] [0].setEnabled (false);
        button [1] [1].setEnabled (false);
        button [1] [2].setEnabled (false);
    }
    if (b [2] [0] != 'b' && b [2] [0] == b [2] [1] && b [2] [1] == b [2] [2]){
        String w = b [2] [0] + " has won!!";
        JOptionPane.showMessageDialog (null, w, "Horizontal Win", JOptionPane.ERROR_MESSAGE);
        button [2] [0].setEnabled (false);
        button [2] [1].setEnabled (false);
        button [2] [2].setEnabled (false);
    }
    if (b [0] [0] != 'b' && b [0] [0] == b [1] [0] && b [1] [0] == b [2] [0]){
        String w = b [0] [0] + " has won!!";
        JOptionPane.showMessageDialog (null, w, "Vertical Win", JOptionPane.ERROR_MESSAGE);
        button [0] [0].setEnabled (false);
        button [1] [0].setEnabled (false);
        button [2] [0].setEnabled (false);
    }
    if (b [0] [1] != 'b' && b [0] [1] == b [1] [1] && b [1] [1] == b [2] [1]) {
        String w = b [0] [1] + " has won!!";
        JOptionPane.showMessageDialog (null, w, "Vertical Win", JOptionPane.ERROR_MESSAGE);
        button [0] [1].setEnabled (false);
        button [1] [1].setEnabled (false);
        button [2] [1].setEnabled (false);
    }
    if (b [0] [2] != 'b' && b [0] [2] == b [1] [2] && b [1] [2] == b [2] [2]){
        String w = b [0] [2] + " has won!!";
        JOptionPane.showMessageDialog (null, w, "Vertical Win", JOptionPane.ERROR_MESSAGE);
        button [0] [2].setEnabled (false);
        button [1] [2].setEnabled (false);
        button [2] [2].setEnabled (false);
    }
    if (b [0] [0] != 'b' && b [0] [0] == b [1] [1] && b [1] [1] == b [2] [2]) {
        String w = b [0] [0] + " has won!!";
        JOptionPane.showMessageDialog (null, w, "Diagonal Win", JOptionPane.ERROR_MESSAGE);
        button [0] [0].setEnabled (false);
        button [1] [1].setEnabled (false);
        button [2] [2].setEnabled (false);
    }
    if (b [0] [2] != 'b' && b [0] [2] == b [1] [1] && b [1] [1] == b [2] [0]) {
        String w = b [0] [2] + " has won!!";
        JOptionPane.showMessageDialog (null, w, "Diagonal Win", JOptionPane.ERROR_MESSAGE);
        button [0] [2].setEnabled (false);
        button [1] [1].setEnabled (false);
        button [2] [0].setEnabled (false);
    }
}
```

(a) Circle 8 sets of repeated code in the before section. /2

(b) Write the oneLine method using the parameters given. /7

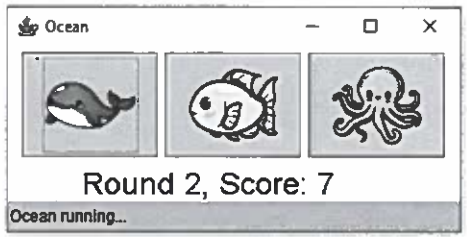
```
public void oneLine(int x1, int y1, int x2, int y2, int x3, int y3, String dir)
{
if (b[ x1 ][ y1 ] != 'b' &&
   b[ x1 ][ y1 ]==b[ x2 ][ y2 ] && b[ x2 ][ y2 ]==b[ x3 ][ y3 ])
{
    String w = b [ x1 ][ y1 ] + " has won!!";
    JOptionPane.showMessageDialog( null , w ,
                dir , JOptionPane.ERROR_MESSAGE);
    button [ x1 ][ y1 ].setEnabled( false );
    button [ x2 ][ y2 ].setEnabled( false );
    button [ x3 ][ y3 ].setEnabled( false );
  }
}
```

(c) Rewrite the win method to call the oneLine method (coded in part c). /6

```
public void win(){
  oneLine( 0 , 0 , 0 , 1 , 0 , 2 , "Horizontal Win" );
  oneLine( 1 , 0 , 1 , 1 , 1 , 2 , "Horizontal Win" );
  oneLine( 2 , 0 , 2 , 1 , 2 , 2 , "Horizontal Win" );
  oneLine( 0 , 0 , 1 , 0 , 2 , 0 , "Vertical Win" );
  oneLine( 0 , 1 , 1 , 1 , 2 , 1 , "Vertical Win" );
  oneLine( 0 , 2 , 1 , 2 , 2 , 2 , "Vertical Win" );
  oneLine( 0 , 0 , 1 , 1 , 2 , 2 , "Diagonal Win" );
  oneLine( 0 , 2 , 1 , 1 , 2 , 0 , "Diagonal Win" );
}
```

This is NOT part of the sample test. You are going to ask me for a second question to practice. This is the second question.

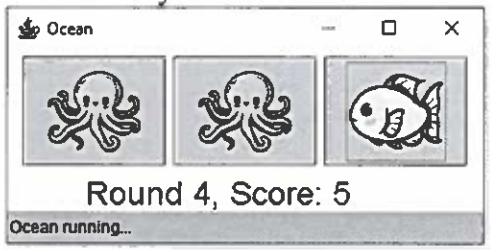
0. In the Ocean Game, we find yet another completely ridiculous situation generated by Ms. Gorski to test your ability to make methods from repeated code… oops, I mean, the entirely realistic "Ocean Game" is played as follows:



Three types of animals are swimming around. You are an oceanographer and dip your net into the water. Depending on what you catch, you get a different number of points.

Animal	Fish	Octopus	Whale
Picture			
Pic Name	f.png	o.png	w.png
Points	1	2	4

To "dip their net", the oceanographer clicks on any of the buttons.



Originally, it was coded like this:

```
public void actionPerformed (ActionEvent e)
{
   int total = 0;
   round++;

   int n = (int) (Math.random () * 3);
   if (n == 0)
      d [0] = 'f';
   else if (n == 1)
      d [0] = 'o';
   else
      d [0] = 'w';
   a.setIcon (createImageIcon (d [0] + ".png"));
   if (d [0] == 'w')
      total += 4;
   else if (d [0] == 'o')
      total += 2;
   else
      total += 1;

   n = (int) (Math.random () * 3);
   if (n == 0)
      d [1] = 'f';
   else if (n == 1)
      d [1] = 'o';
   else
      d [1] = 'w';
   b.setIcon (createImageIcon (d [1] + ".png"));
   if (d [1] == 'w')
      total += 4;
   else if (d [1] == 'o')
      total += 2;
   else
      total += 1;

   n = (int) (Math.random () * 3);
   if (n == 0)
      d [2] = 'f';
   else if (n == 1)
      d [2] = 'o';
   else
      d [2] = 'w';
   c.setIcon (createImageIcon (d [2] + ".png"));
   if (d [2] == 'w')
      total += 4;
   else if (d [2] == 'o')
      total += 2;
   else
      total += 1;

   output.setText ("Round " + round + ",
                    Score: " + total);
}
```

(a) ☐ In the previous column, put a square around three sets of repeated code. /0

(b) Fill in the change method to alter the pictures on the screen. /0

```
public void change (JButton sq, int pos) {
    int n = (int) (Math.random () * 3);
    if (n == 0)
        d [pos] = 'f';
    else if (n == 1)
        d [pos] = 'o';
    else
        d [pos] = 'w';
    sq.setIcon (createImageIcon (d[pos]+".png"));
}
```

(c) Fill in the points method to return the right number of points, based on the picture stored in the d array. /0

```
public int points (int n) {
    if (d [n] == 'f')
        return 4;
    else if (d [n] == 'o')
        return 2;
    else
        return 1;
}
```

(d) Fill in the actionPerformed to call the above two methods. /0

```
public void actionPerformed (ActionEvent e) {
    int total = 0;
    round++;
    change (a, 0);
    total += points (0);
    change (b, 1);
    total += points (1);
    change (c, 2);
    total += points (2);
    output.setText ("Round " + round + ",
                                Score: " + total);
}
```