

Unit 2 – ICS4U0 – Methods and Recursion

Sample Test – Wednesday October 7, 2026

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

Riddle Answer: tongue

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

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

double price = 3.14159 ;
Param Type Param Name Initial Value:
int ans = jet (price);
Return Type Variable Name Method Name (Param Name):

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

double ans = glider ();
Param Type Param Name Initial Value:
double ans = glider ();
Return Type Variable Name Method Name (Param Name):

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

char pilot = 'a' ;
Param Type Param Name Initial Value:
String ans = drone (pilot);
Return Type Variable Name Method Name (Param Name):

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

int control = 5 ;
Param Type Param Name Initial Value:
void helicopter (control);
Return Type Variable Name Method Name (Param Name):

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 (circled around the if block)
Recursive Case (circled around the else block)

(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

5. Provide the phrase or number required. /10

120
Recursive
Loop
112358
* probability calculations
* extensibility
* rabbit populations
Return type
stack overflow
base case

- (a) The answer to the 5! (five factorial)
- (b) A method that calls itself.
- (c) Every recursive method can be re-coded as this.
- (d) The first 6 terms of the Fibonacci sequence.
- (e) What are factorials used for in math?
- (f) By putting code in one place, changes are easier. (ORATE)
- (g) Where does the Fibonacci sequence appear in nature?
- (h) The name of the output of a method.
- (i) The equivalent of an infinite loop in recursion.
- (j) The equivalent of a loop stopping condition in recursion.

**can differ*

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

Reusability, instead of copy/pasting code, call the method, best applies to the Pizza Party Program. Before we changed there were 5 ifs, one to "roll" each plate. After we created a rollBig method, all the plates reused one method's code.

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

(see sheet 2.5) The parameters get smaller and smaller in the recursive case - the call is something like seq(n-1). Eventually, the parameter will be small enough that the base case is reached (eg. n==1) and a value can be returned and the recursion stops.

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);
}
```

you can go right to the answer if you want but I can give you part marks if you show work :-)

wing(-1)	wing(0)	wing(1)	wing(2)	wing(3)	wing(4)	wing(5)
2	2	2	$= \text{wing}(1) + \text{wing}(-1)$ $= 2 + 2$ $= 4$	$= \text{wing}(2) + \text{wing}(0)$ $= 4 + 2$ $= 6$	$= \text{wing}(3) + \text{wing}(1)$ $= 6 + 2$ $= 8$	$= \text{wing}(4) + \text{wing}(2)$ $= 8 + 4$ $= 12$

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

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

/6

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

must match can differ also must match

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

/5

```
public void endTurn (int 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);
}
```

an extra error to help you out.

3

Application

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

/7

(a) Fill in the setPic method:

```
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 );
}
```

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;
}
```

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[2][0] != 'b' && b[2][0] == b[1][1] && b[1][1] == b[0][2]) {
        String w = b[2][0] + " has won!";
        JOptionPane.showMessageDialog (null, w, "Diagonal Win", JOptionPane.ERROR_MESSAGE);
        button[2][0].setEnabled (false);
        button[1][1].setEnabled (false);
        button[0][2].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 () {

```

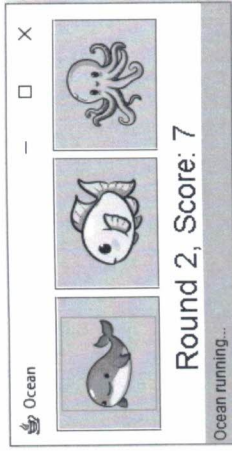
```

    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.

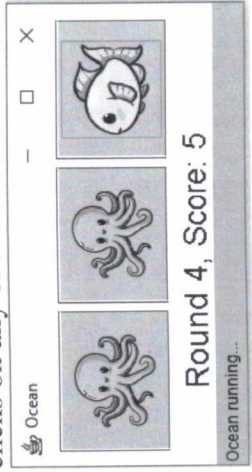
0. In the Ocean Game, we find yet another completely ridiculous situation generated by Ms. Gorski to test your ability to make methods from repeated code... oops, I mean, the entirely realistic "Ocean Game" is played as follows:



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[0] = 'f';
    else if (n == 1)
        d[0] = 'o';
    else
        d[0] = 'w';
    b.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[0] = 'f';
    else if (n == 1)
        d[0] = 'o';
    else
        d[0] = 'w';
    c.setIcon (createImageIcon (d[0] + ".png"));
    if (d[0] == 'w')
        total += 4;
    else if (d[0] == '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.

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

```
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.

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

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

```
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);
}
```