



ICS4U **Sample** Final Exam

Computer Science

Monday January 26, 2026

Period 3 Exam Day, Room 129

Ms. Gorski

7 Question Pages

18 Questions

School Time Allotted: 8:30 – 11:00

Expected Time to Write Exam: 70 minutes

Name (first and last): _____

Signature: _____

Total: / **140** %

Instructions:

- Students should answer all questions directly on the exam paper.
- Students may write in pencil or black pen or dark blue pen.
- Students will have access to the Objects Memory Aid as used on their Objects Test.

- Extra paper is permitted, but probably won't be needed.
- Water is permitted.
- Additional materials, such as bags, coats, and phones, are not permitted at the student's desk during the exam. They may be left at home, or in the student's locker, or at the front of the room.
- No phones or other electronic devices are permitted. Smart watches, headphones and earbuds are not permitted at the student's desk or in their pockets. These devices must be left in the student's bag at the front of the room.

1. This is the original String:
String sm = "Super Mario";

0	1	2	3	4	5	6	7	8	9	10

/5

Match the code with the output.

u	a. System.out.println (sm.substring (0, 3));
R	b. System.out.println (sm.charAt (1));
true	c. System.out.println ((int) sm.charAt (7));
false	d. System.out.println (sm.indexOf ('r'));
Sup	e. System.out.println (sm.indexOf ('u'));
mar	f. System.out.println (sm.compareTo ("Pong") < 0);
iou	g. System.out.println (sm.equals ("Super Mario"));
97	h. System.out.println (sm.length ());
4	i. System.out.println (sm.toLowerCase ());
11	j. System.out.println (sm.toUpperCase ());
1	k. System.out.println (sm.replace ('r', 'y'));
Supey Mayio	l. System.out.println (sm.toUpperCase().charAt (8));
SUPER MARIO	m. System.out.println (" " + sm.toLowerCase().substring(6,9));
super mario	n. System.out.println (sm.substring (9, 11) + sm.charAt (1));

2. This is the original String: String y = "a yellow wood door";

/10

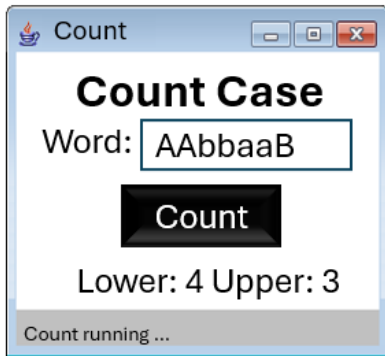
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Write the code to produce the following output:

(a)	a yezzow wood door	System.out.println() ;
(b)	A YELLOW WOOD DOOR	System.out.println() ;
(c)	r	System.out.println() ;
(d)	6 //position of o	System.out.println() ;
(e)	yellow	System.out.println() ;
(f)	18 //the length	System.out.println() ;
(g)	97 //ASCII of a	System.out.println() ;
(h)	odd	System.out.println() ;
(i)	a yellow m moo f foor	System.out.println() ;
(j)	weed deer	System.out.println() ;

3. White Box and Black Box test this app.

/9



This app counts the number of lowercase and uppercase letters in the word that is entered.

Word	Black Box Test Case
Word	White Box Test Case & Line Number

```

1  public void actionPerformed (ActionEvent e){
2      String s = input.getText();
3      int small = 0;
4      int big = 0;
5      for (int i = 0 ; i < s.length () ; i++) {
6          if (((int) s.charAt (i)) >= 97)
7              small++;
8          else
9              big++;
10     }
11     result.setText ("Lower: " + small + " Upper: " + big);
12 }

```

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;
}

```

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

Evaluate the following:

(f) mandlebrot(1).

(g) mandlebrot(4).

(h) mandlebrot(0).

5. 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.

_____ (_____) {
 method start word return type (output) method name (whatever) parameter type parameter name (input)

- (a) ☐ Circle the three sets of repeated code in the original actionPerformed.



The picture names, the corresponding random numbers.



```
public void actionPerformed(ActionEvent e) {
    int r1 = (int) Math.random()*4;
    if (r1 == 0)
        f1.setIcon(createImageIcon("ski.png"));
    else if (r1 == 1)
        f1.setIcon(createImageIcon("scarf.png"));
    else if (r1 == 2)
        f1.setIcon(createImageIcon("skate.png"));
    else
        f1.setIcon(createImageIcon("gloves.png"));
    int r2 = (int) Math.random()*4;
    if (r2 == 0)
        f2.setIcon(createImageIcon("ski.png"));
    else if (r2 == 1)
        f2.setIcon(createImageIcon("scarf.png"));
    else if (r2 == 2)
        f2.setIcon(createImageIcon("skate.png"));
    else
        f2.setIcon(createImageIcon("gloves.png"));
    int r3 = (int) Math.random()*4;
    if (r3 == 0)
        f3.setIcon(createImageIcon("ski.png"));
    else if (r3 == 1)
        f3.setIcon(createImageIcon("scarf.png"));
    else if (r3 == 2)
        f3.setIcon(createImageIcon("skate.png"));
    else
        f3.setIcon(createImageIcon("gloves.png"));

    calculateWin(r1, r2, r3);
    updateWinMessage();
}
```

- (b) Create a setPic method to replace the repeated code:

```
public void setpic(_____, int num) {
    if (_____ == _____)
        _____.setIcon(createImageIcon("_____"));
    else if (_____ == _____)
        _____.setIcon(createImageIcon("_____"));
    else if (_____ == _____)
        _____.setIcon(createImageIcon("_____"));
    else
        _____.setIcon(createImageIcon("_____"));
}
```

- (c) 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:
    _____(_____, _____);
    _____(_____, _____);
    _____(_____, _____);
    calculateWin(r1, r2, r3);
    updateWinMessage();
}
```

7. What does ORATE stand for?

/5

--	--	--	--	--

8. In the first column, fill in the terms that match the description.

/8

	(a) Variables found inside an object's class.
	(b) A template for an object or type. Also contains a java program.
	(c) An object method type that returns values of instance variables.
	(d) An object method type that sets up dynamic memory.
	(e) An object method type that changes the values of instance variables.
	(f) Keeping an object's code self-contained and independent of other code. It relies only on itself.
	(g) A general term for data and the methods associated with that data.
	(h) Other programmers can use your objects without deep understanding of the specifics of how it is coded.

9. Circle the best data structure for each description.

/4

- Stack Queue (a) A FIFO structure.
- Stack Queue (b) A LIFO structure.
- Stack Queue (c) A pile of books model this data structure well.
- Stack Queue (d) The data structure that could model a waiting line.

10. Circle the **sorting** algorithm which is the best choice for each situation.

/4

- Bubble Quick Merge Selection (a) The list of names is sorted; you add one element to the front. You have extra memory.
- Bubble Quick Merge Selection (b) The array not randomized; it is very large. You just have enough memory to hold it.
- Bubble Quick Merge Selection (c) The char array is in reverse order, but it is only 62 elements long.
- Bubble Quick Merge Selection (d) You have lots of extra memory and the double array is in random order.

11. Circle the **search** which is the best choice for each situation.

/4

- Binary Linear (a) You have a list of heights, ordered from smallest to greatest.
- Binary Linear (b) You are looking up a word in a dictionary.
- Binary Linear (c) You look up a definition in a dictionary to find the word that goes with it.
- Binary Linear (d) You are looking up a student ID in a list with no apparent order.

12. Put these algorithm speeds in order.
(1 is fastest, 6 is slowest)

- ___ $O(n)$
- ___ $O(n!)$
- ___ $O(\log n)$
- ___ Constant time
- ___ $O(n \log n)$
- ___ $O(n^2)$

13. What speed is each algorithm?

/10

- (a) Swap _____
- (b) Quicksort _____
- (c) Binary Search _____
- (d) Stack Pop _____
- (e) BogoSort _____
- (f) Linear Search _____
- (g) Queue Dequeue _____

14. (a) Fill in the Tile class (instance variables: char letter, int points) that holds data for a scrabble tile.

/20

```
// Class line
public _____ {
    // Instance Variables
    _____;
    _____;

    // Constructors
    public _____ () {
        _____ = _____;
        _____ = _____;
    }
    public _____ (_____, _____) {
        _____ = _____;
        _____ = _____;
    }

    // Mutator
    public _____ (_____)
    {
        _____ = _____;
    }
    public _____ (_____)
    {
        _____ = _____;
    }

    // Accessor
    public _____ (_____)
    {
        _____;
    }
    public _____ (_____)
    {
        _____;
    }

    // To String (Accessor)
    public _____ (_____)
    {
        _____ " " + _____
        + " " + _____;
    }

    // CompareTo, use letter
    public _____ (_____)
    {
        if (_____ > _____._____)
            return _____;
        else if (_____ < _____._____)
            return _____;
        else
            return _____;
    }
}
```



```
//Returns triple the points
public _____ (_____) {
    return _____;
}

// Equals (Facilitator) use points
public _____ (_____)
{
    if (_____ == _____._____)
        return _____;
    else
        return _____;
}
}
```



(b) Adapt the Stack Class to make a Stack of Tiles.

```
public class _____ Stack {
    private int count;

    private Object data[] = new Object [50];

    public _____ Stack () {
        count = 0; }

    public void push (Object addMe) {
        data [count] = addMe;
        count++; }

    public int size () {
        return count; }

    public boolean isFull () {
        return (count == 50); }

    public Object pop () {
        count--;
        return data [count]; }

    public Object peek () {
        return data [count--]; }

    public boolean isEmpty () {
        return count == 0; }

    public void clear () {
        count = 0; }
}
```

(c) Create a new Stack of Tiles, add two Tiles to it.

```
_____ Stack s = new _____ Stack();

Tile q = new Tile(_____, _____);
s.push(____);

Tile d = new Tile(_____, _____);
s.push(____);
```

15. Using the pictures shown, fill in the code for this game of Cookie Run.

/5

The picture information:



The code:

```
int oven [][]={ { ____, ____, ____, ____, ____},
                 { ____, ____, ____, ____, ____},
                 { ____, ____, ____, ____, ____},
                 { ____, ____, ____, ____, ____} };
```

The grid information:



```
int row = ____;  
int col = ____;  
JButton pics[]=new JButton[____*____];  
Panel grid=new Panel(new GridLayout(____,____));  
int m=0;  
for(int i=0; i<____; i++){  
    for(int j=0; j<____; j++){  
        pics[m]=new JButton  
            (createImageIcon(____[i][j]+".jpg"));  
        pics[m].addActionListener(this);  
        pics[m].setActionCommand(m + "");  
        grid.add(pics[m]);  
        m++;  
    }  
}  
add(grid);
```

16. Fill in the neighbours chart for the center element .

/5

field[_____] [_____]	field[_____] [_____]	field[_____] [_____]
field[_____] [_____]	Clicked Element field[i] [j]	field[_____] [_____]
field[_____] [_____]	field[_____] [_____]	field[_____] [_____]

17. In the game Cookie Run, you get points every time you get a GingerBrave (6) to the left OR to the right of a balloon (3). In the screen shown, the user would get four points; the bottom GingerBrave would count on both sides.

/8



Code a method that returns the number of GingerBraves (6) to the right or left of a balloon (3) in a global array named `oven`. The variables `row` and `col` track the dimensions for the array. The method has no parameters but will return an integer.

```

public _____ (_____) {
    _____ = _____;
    for (_____) {
        for (_____) {
            if(_____ && field[____][____]==__&& field[____][____]==__)
                _____;
            else if(_____ && field[____][____]==__&& field[____][____]==__)
                _____;
        }
    }
    _____;
}

```

🐼

🎈

🍪

🍪

🍪

1
2
3
4
5



