

Final Exam

● Graded

Student

AKSHAJ KAMMARI

Total Points

112.5 / 135 pts

Question 1

Rutgers Honor Pledge

0 / 0 pts

✓ + 0 pts Correct

+ 0 pts Incorrect

Question 2

Arrays & Functions

25 / 25 pts

2.1 (no title)

5 / 5 pts

✓ + 5 pts Correct

Answer: 11 **14** 7 12 **15** 69 42 **45** 39 - (2 pts for non-bolded items - 3 pts for bolded items)

For ALL bolded or non bolded items

+ 0 pts Incorrect

✓ + 20 pts Correct

Solution:

```
public static int generateEven(){
    int randEven = (int)(Math.random() * 50) * 2;
    return randEven;
}

public static int generateOdd(){
    int randOdd = ((int)(Math.random() * 50) * 2) + 1;
    return randOdd;
}

public static String[] nextRound(int[] a, String[] n){
    String[] s = new String[a.length/2];
    int count = 0;
    for(int i = 0; i < a.length; i += 2){
        if (a[i] > a[i+1]){
            s[count] = n[i];
        }
        else{
            s[count] = n[i + 1];
        }
        count++;
    }
    return s;
}
```

(5 pts) generateEven

1 pt for returning a generated random int.

3 pts for generating an even int

1 pt for the correct range of even ints.

(5 pts) generateOdd

1 pt for returning a generated random int.

3 pts for generating an odd int

1 pt for the correct range of odd ints.

(10 pts) nextRound

2 pts for creating new array that is

+ 0 pts incorrect

Question 3

Recursion

12 / 30 pts

3.1 (no title)

12 / 12 pts

✓ + 12 pts Correct

- a. (3 pts) What is the output if `foo(2, 0)` is called?
a. ANSWER. 1
- b. (3 pts) What is the output if `foo(3,2)` is called?
a. ANSWER. 9
- c. (3 pts) What is the output if `foo(2,3)` is called?
a. ANSWER. 8
- d. (3 pts) Explain what `foo` does? That is, what does it compute given `x` and `y`?
a. The code computes the power function $\text{foo}(x,y) = x^y$

+ 3 pts A Correct

+ 3 pts B Correct

+ 3 pts C Correct

+ 3 pts D Correct

✓ + 0 pts Incorrect

3.2 (no title)

0 / 6 pts

+ 6 pts Correct

$$\begin{aligned}
 (x1, y1) &= \text{____}(x - d/2, y - d/2)\text{____} \\
 (x2, y2) &= \text{____}(x + d/2, y - d/2)\text{____} \\
 (x3, y3) &= \text{____}(x + d/2, y + d/2)\text{____} \\
 (x4, y4) &= \text{____}(x - d/2, y + d/2)\text{____}
 \end{aligned}$$

+ 1.5 pts A correct

+ 1.5 pts B correct

+ 1.5 pts C correct

+ 1.5 pts D correct

✓ + 0 pts Incorrect

+ 12 pts

```
(i)    draw( n-1 , x + d/2 , y - d/2 , d/2 );  
(ii)   draw( n-1 , x + d/2 , y + d/2 , d/2 );  
(iii)  draw( n-1 , x - d/2 , y - d/2 , d/2 );  
(iv)   draw( n-1 , x - d/2 , y + d/2 , d/2 );
```

or

```
draw(n-1, x+d/2, y+d/2, d/2);  
draw(n-1, x+d/2, y-d/2, d/2);  
draw(n-1, x-d/2, y+d/2, d/2);  
draw(n-1, x-d/2, y-d/2, d/2);
```

Correct

✓ + 0 pts incorrect

+ 6 pts Incorrect order

Question 4

OOP

40 / 40 pts

✓ + 16 pts Correct

```
public class RaceCar{
    private int currentFuel; //in gallons
    private int currentSpeed; //in mph
    private int speedLimit;
    private int maxFuelCapacity;

    public RaceCar(int currentFuel, int currentSpeed, int speedLimit, int maxFuelCapacity){
        //implement this method
        this.currentFuel=currentFuel;
        this.currentSpeed=currentSpeed;
        this.speedLimit=speedLimit;
        this.maxFuelCapacity=maxFuelCapacity;
    }

    public boolean increaseSpeed(int speedFactor){
        //implement this method. Return true if speed was successfully increased, false
        //otherwise
        if(currentSpeed+speedFactor<=speedLimit){
            currentSpeed+=speedFactor;
            return true;
        }
        return false;
    }

    public boolean decrease(int speedFactor){
        //implement this method. Return true if speed was successfully decreased, false
        //otherwise
        if(currentSpeed-speedFactor>=0){
            currentSpeed-=speedFactor;
            return true;
        }
        return false;
    }

    public boolean increaseFuel(int fuel){
        //implement this method. Return true if fuel was successfully added, false otherwise
        if(currentFuel+fuel<=maxFuelCapacity){
            currentFuel+=fuel;
            return true;
        }
        return false;
    }

    public boolean decreaseFuel(int fuel){
        //implement this method. Return true if fuel was successfully decreased, false
        //otherwise
        if(currentFuel-fuel>=0){
            currentFuel-=fuel;
            return true;
        }
        return false;
    }

    public void emptyFuel(){
```



```
        //implement this method to deplete the tank of all fuel
        this.currentFuel=0;
    }

    public int getSpeed(){
        //return the current speed of the race car
        return this.currentSpeed;
    }

    public int getFuel(){
        //return the current amount of fuel in the race car
        return this.currentFuel;
    }

}
```

+ 0 pts incorrect

4.2 (no title)

4 / 4 pts

✓ + 4 pts Correct

+ 0 pts Incorrect

✓ + 20 pts Correct

```
public class myRect
{
    // A class to keep a myRectangle
    private int width;
    private int height;

    public myRect()
    {
        this.width = 1;
        this.height = 1;
    }

    public myRect(int side)
    {
        this.width = side;
        this.height = side;
    }

    public myRect(int w, int h)
```

```
{
    this.width = w;
    this.height = h;
}

public int getWidth()
{
    return this.width;
}

public int getHeight()
{
    return this.height;
}
```

```
public int perimeter()
{
    return (2*this.width)+(2*this.height);
}

public int area()
{
    return (this.width * this.height);
}

public String toString()
{
    return "A " + this.width + "X" + this.height + " rectangle";
}

// EOC

// test code
public static void main(String[] args)
{
    StdOut.println("My Rectangle");
    myRect R1 = new myRect();
    myRect R2 = new myRect(5);
    myRect R3 = new myRect(3,4);

    StdOut.println(R1.getWidth());
    StdOut.println(R2.area());
    StdOut.print(R3);
}
}
```

+ 0 pts incorrect

+ 0 pts [Click here to replace this description.](#)

Question 5

Sorting & Searching

17.5 / 20 pts

5.1 (no title) 2.5 / 2.5 pts

✓ + 2.5 pts Correct = n-1 times

+ 0 pts Incorrect

5.2 (no title) 2.5 / 2.5 pts

✓ + 2.5 pts Correct = 31

+ 0 pts Incorrect

5.3 (no title) 5 / 5 pts

✓ + 5 pts Correct = A. The array is divided into two sorted regions of roughly $n/2$ items. If the array has an odd number of items these two regions have sizes $n/2$ and $n/2-1$ respectively and B. [2, 5, 9, 16, 1, 4, 17]

+ 2.5 pts A is correct

+ 2.5 pts B is correct

+ 0 pts Incorrect

5.4 (no title) 5 / 5 pts

✓ + 5 pts Correct = C

+ 0 pts Incorrect

5.5 (no title) 0 / 2.5 pts

+ 2.5 pts Correct = D

✓ + 0 pts Incorrect

5.6 (no title) 2.5 / 2.5 pts

✓ + 2.5 pts Correct = A

+ 0 pts Incorrect

Question 6

Complexity

18 / 20 pts

6.1 (no title)

8 / 10 pts

✓ + 1 pt 1: $O(1)$

+ 1 pt 2: $O(n^2)$

✓ + 1 pt 3: $O(n)$

✓ + 1 pt 4: NA

✓ + 1 pt 5: $O(n)$

✓ + 1 pt 6: $O(n^2)$

✓ + 1 pt 7: $O(n!)$

✓ + 1 pt 8: $O(n^2)$

✓ + 1 pt 9: $O(n)$

+ 1 pt 10: $O(n^2)$

✓ + 0 pts Incorrect

+ 10 pts Correct

$O(1)$, $O(n^2)$, $O(n)$, NA, $O(n)$, $O(n^2)$, $O(n!)$, $O(n^2)$, $O(n)$, $O(n^2)$

1. Find the middle element in an array $O(1)$
2. Determine if an unsorted array has any duplicates $O(n^2)$
3. Determine if a given array is sorted $O(n)$
4. Binary search for a target value in an unsorted array NA
5. Convert an array of size n in descending order to an array in ascending order. That is, $[3\ 2\ 1] \rightarrow [1\ 2\ 3]$ $O(n)$
6. Given a $n \times n$ 2D array of integers, check if there are two elements on the diagonal whose sum is 0. The diagonal elements in a 2D array are given by $a[i][i]$ (uses the same index for row and column) $O(n^2)$
7. Finding all permutations of the n elements in the array A $O(n!)$
8. Sort an array of size n using selection sort $O(n^2)$
9. Find the second largest element in an unsorted array $O(n)$
10. Copy the values from a 2d array to another 2d array (assume they are the same size). $O(n^2)$ (above 1 pt each)

6.2 (no title)

5 / 5 pts

✓ + 5 pts Correct = The complexity of Mergesort is $n \log n$ because Mergesort is two parts, the splits which are $\log n$ for complexity and the merge which is linear as n . In order to get the complexity of the whole sort you need to multiply the two (just like a selection or Insertion sorted list is list x list or n^2) and you get $n \log n$ as a solution

+ 0 pts incorrect

6.3

(no title)

5 / 5 pts

✓ **+ 5 pts** Correct = 16 times(or two doubling experiments where time is 4 times greater when doubling the data)
(5 points)

+ 0 pts incorrect

Q1 Rutgers Honor Pledge

0 Points

Rutgers honor pledge: On my honor, I have neither received nor given any unauthorized assistance during this examination.

You are on your honor to complete this quiz without assistance. You MAY NOT ask any human for assistance or use any online tutoring service.

By continuing, you are AGREEING TO THE ABOVE HONOR STATEMENT and Rutgers Academic Integrity policies.

Q2 Arrays & Functions

25 Points


Q2.1

5 Points

What will be printed as a result of executing the following Java program?

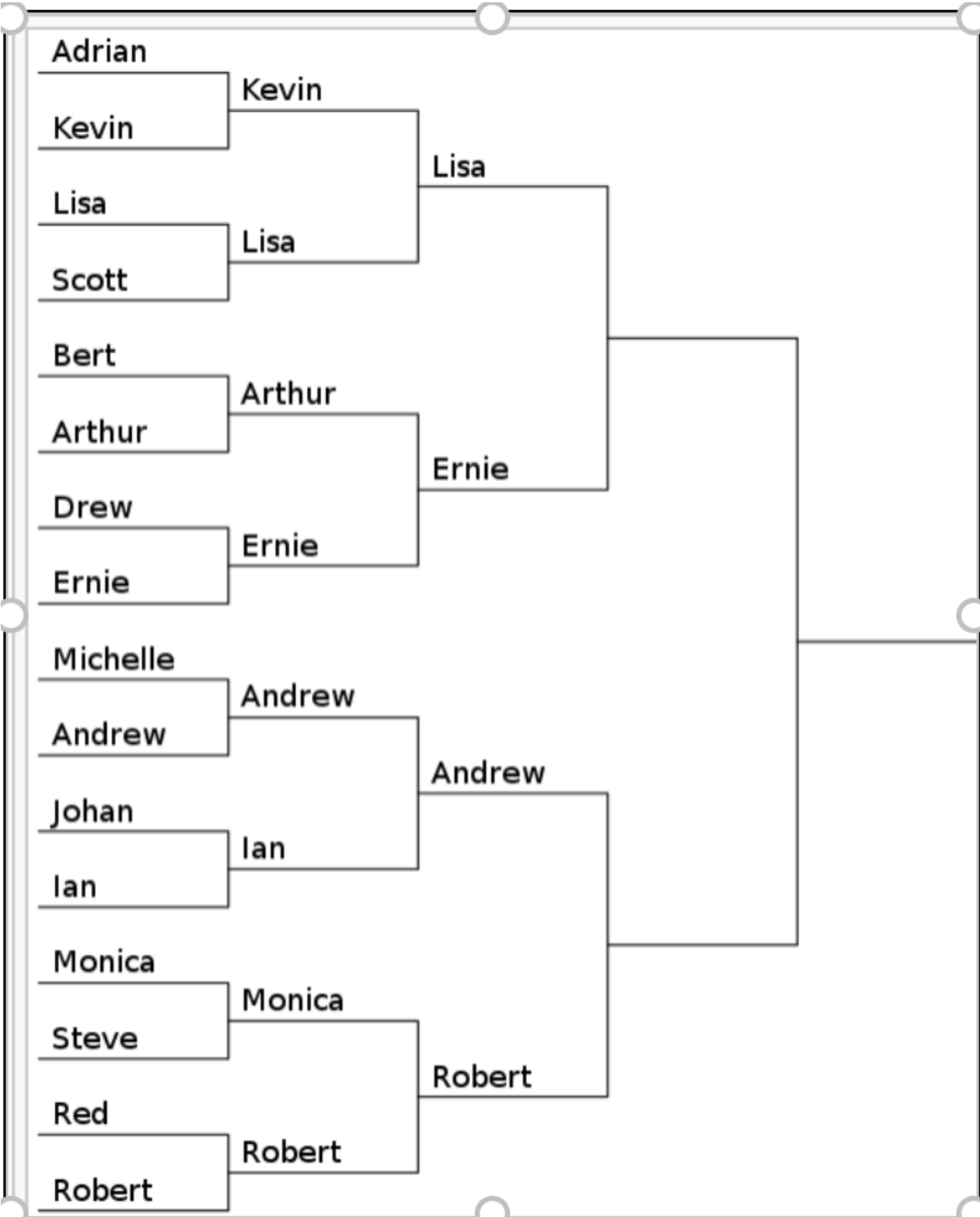
```
public class ProgramFinal{  
    Run | Debug  
    public static void main(String[] args) {  
        int[] b = {11, 20, 7, 12, 15, 69, 42, 21, 39};  
        mystery2(3, b);  
        printArray(b);  
    }  
    public static void printArray(int[] b){  
        for(int elt : b){  
            System.out.print(elt + " ");  
        }  
        System.out.println();  
    }  
    public static void mystery2(int x, int[] nums){  
        for (int k = 1; k < nums.length; k = k + x){  
            nums[k] = nums[k - 1] + x;  
        }  
    }  
}
```

11 14 7 12 15 69 42 45 39

 No files uploaded

Q2.2
20 Points

A FooGame Tournament is a single elimination tournament. Once a player loses, the player is out. Each round has half as many players as the round before. The rounds continue until there is ONE winner.



In our simulated tournament:

- a String array is filled with player names. An int array is filled with player scores such that names[0] has their score recorded in scores[0], names[1] has his/her score recorded in scores[1], etc.
- There are 16 players in the tournament. The number of players is a power of 2 input by the user.
- The player scores are randomly generated such that the players in the even positions have even scores (0 is considered even), and the players in the odd positions have odd scores.
- Player 0 plays against Player 1, Player 2 plays against Player 3, etc.
- The player with the higher of the two scores advances to the next round. In the diagram above, Kevin had a higher score than Adrian and therefore advanced to the next round. The incomplete program code is below.
- The game is mostly written for you. You are tasked with writing **THREE** methods:
 - **(5 points) generateEven:** has no parameters; generates and returns a random **even** integer in the range 0 – 98 inclusive.
 - **(5 points) generateOdd:** has no parameters; generates and returns a random **odd** integer in the range 1 – 99 inclusive.
 - **(10 points) nextRound:** This method has two parameters: a String array, **players**, containing the names of players and an int array, **theScores**, containing the scores of the players such that players[0] has score theScores[0], players[1] has score theScores[1], etc. This method returns a String array containing the winners of the tournament games for that round. The array that is returned is half the size of the parameter array. The tournament pairings are consecutive array elements (i.e. players[0] and players[1] play against each other; players[2] and players[3] play against each other, etc.

The program code is below.

The program code is below.

```
public class Game{
    public static void main(String[] args){
        int numPlayers = Integer.parseInt(args[0]); // This number is a power of 2 input by the user.
        String[] names = new String[numPlayers];
        for(int k = 0; k < names.length; k++){
            names[k] = StdIn.readString(); // Player names
        }

        while (names.length > 1){
            System.out.println("\nCurrent round of players:");
            int[] scores = new int[names.length];
            for (int i = 0; i < names.length; i++){
                if (i%2 == 0){
                    scores[i] = generateEven();
                }else{
                    scores[i] = generateOdd();
                }
            }
            for (int j = 0; j < names.length; j++){
                System.out.println(names[j] + "\t\t" + scores[j]);
            }
            names = nextRound(scores,names);
        }

        System.out.println(names[0] + " is the winner!");
    }
}
```

The game: Each iteration of the while loop will generate the scores of the players in a round of the tournament. Each round has half as many players. You are to write the code for the three methods:

- generateEven
- generateOdd
- nextRound

```
// Generates and returns an even integer
// in the range of 0 to 98 inclusive.
```

```
public static int generateEven() {
```

```
    //You need to write this code
```

```
}
```

```
// Generates and returns an odd integer
```

```
// in the range of 1 to 99 inclusive.
```

```
public static int generateOdd() {
```

```
    //You need to write this code
```

```
}
```

```
/* This method has two parameters: a String array,
   players, containing the names of players and an
   int array, theScores, containing the scores of
   the players such that players[0] has score
   theScores[0], players[1] has score
   theScores[1], etc.
```

This method returns a String array containing the winners of the tournament games for that round. The array that is returned is half the size of the parameter array. The tournament pairings are consecutive array elements (i.e. players[0] and players[1] play against each other; players[2] and players[3] play against each other, etc.

```
*/
```

```
public static String[] nextRound(int[] theScores,
    String[] players) {
```

```
    //You need to write this code
```

```
}
```

The command **java Game 16 < players.txt** may result with the following display:

Current round of players:

Adrian	12	
Kevin	57	
Lisa		36
Scott	11	
Bert		64
Arthur	67	
Drew	38	
Ernie	31	
Mike	42	
Andrew	99	
Johan	38	
Ian		73
Monica	62	
Steve	15	
Red	54	
Robert	9	

Current round of players:

Kevin	66	
Lisa		43
Arthur	22	
Drew	87	
Andrew	32	
Ian		91
Monica	70	
Red	1	

Current round of players:

Kevin	62	
Drew	21	
Ian		12
Monica	53	

Current round of players:

Kevin	30	
Monica	83	

Monica is the winner!

```
// Generates and returns an even integer
// in the range of 0 to 98 inclusive.
```

```
public static int generateEven() {
int random = (int)(Math.random()(99));
```

```
while(random%2 != 0) {
random = (int)(Math.random()(99));
}
return random;
}
```

```
// Generates and returns an odd integer
```

```
// in the range of 1 to 99 inclusive.
```

```
public static int generateOdd() {  
    int random = (int)(Math.random()(100)+1);
```

```
    while (random%2 != 1) {  
        random = (int)(Math.random()(100)+1);  
    }  
    return random;  
}
```

```
/* This method has two parameters: a String array,  
   players, containing the names of players and an  
   int array, theScores, containing the scores of  
   the players such that players[0] has score  
   theScores[0], players[1] has score  
   theScores[1], etc.
```


This method returns a String array containing the winners of the tournament games for that round. The array that is returned is half the size of the parameter array. The tournament pairings are consecutive array elements (i.e. players[0] and players[1] play against each other; players[2] and players[3] play against each other, etc.

```
*/
```

```
public static String[] nextRound(int[] theScores,  
    String[] players) {  
    String[] Str = new String [players.length/2];  
    int count = 0;
```

```
    for(int i = 0; i < players.length - 1; i += 2) {  
        if(theScores[i] < theScores[i++]) {  
            Str[count] = players[i++];  
        }  
        else {  
            Str[count] = players[i];  
            count++;
```

```
}  
}  
return Str;  
}
```

 No files uploaded

Q3 Recursion

30 Points

Q3.1

12 Points

Consider the recursive function foo GIVEN below:

```
foo(int x, int y) {  
    if (y is 0) return 1;  
    if (y is even) return foo(x*x, y/2);  
    else return foo(x*x, y/2) * x;  
}
```

A. (3 points) What is the output if `foo(2, 0)` is called?

1

B. (3 points) What is the output if `foo(3,2)` is called?

9

C. (3 points) What is the output if `foo(2,3)` is called?

8

D. (3 points) Explain what foo does? That is, what does it compute given x and y?

Foo checks if y is even or not and if it isn't then x is multiplied to the return statement. It computes x^y .

Q3.2

6 Points

H-Tree Part 1

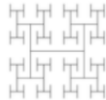
Define a H-tree of order n as a combination of one H-tree of order n and 4 H-trees of order $(n-1)$ etc connected to each tip of H-tree of order n . Each 'H' in a H-tree is drawn by connecting 3 lines of equal length. The H-trees of order $(n-1)$ are half the size of H-tree of order n . As an example, here are the H-trees of orders 1, 2, 3, 4, and 5. H-tree of order 0 is null.



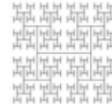
Order 1



order 2



order 3

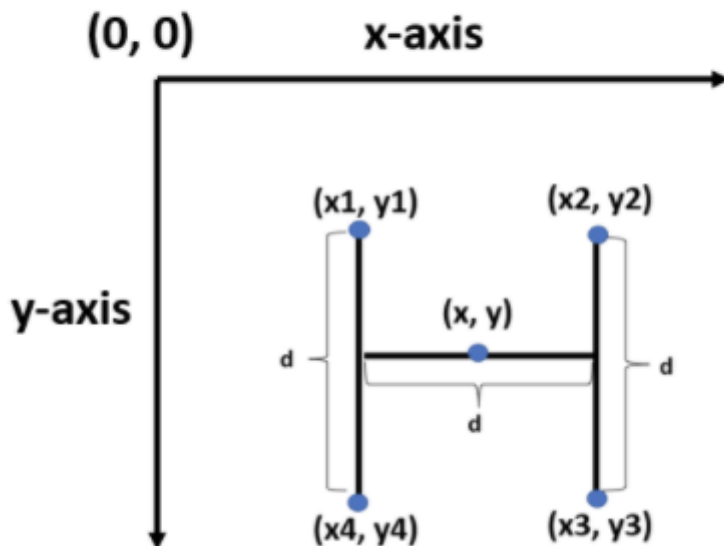


order 4



order5

An H-tree of order 1, centered at (x, y) is shown below. **The length of each side of this tree is d .** Find the coordinates of the 4 tips of the 'H' using (x, y) and d . That is, compute the coordinates (x_1, y_1) , (x_2, y_2) , (x_3, y_3) and (x_4, y_4) using (x, y) and d . Note that the origin $(0, 0)$ of the coordinate system is at upper-left corner.



A. (1.5 points) $(x_1, y_1) =$

$(x-d/2, y+d/2)$

B. (1.5 points) $(x_2, y_2) =$

$(x+d/2, y+d/2)$

C. (1.5 points) $(x_3, y_3) =$

$(x+d/2, y-d/2)$

D. (1.5 points) $(x_4, y_4) =$

$(x-d/2, y-d/2)$

Q3.3

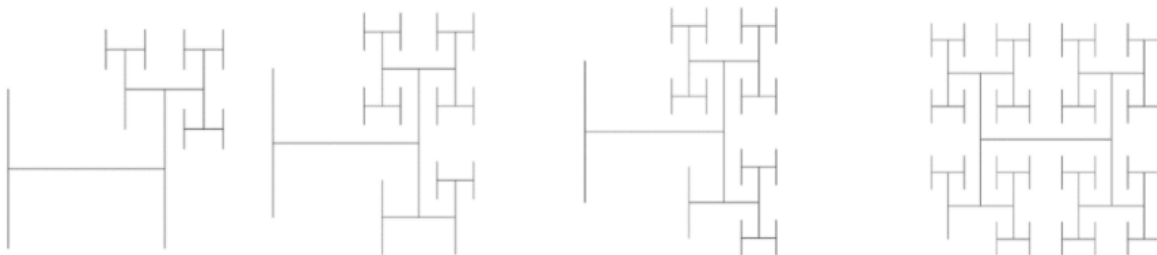
12 Points

H-tree Part 2

Suppose you have implemented a recursive H-tree drawing program using the following code.

```
void draw(int n, double x, double y, double d) {  
    if (n == 0) return;  
    /* code to draw 3 lines to make 'H' that centers at (x, y) with  
    Each line of length equals to d.  
    */  
    void drawH(double x, double y, double d) {  
        /* code to draw a H-tree of order n */  
  
        /* first recursive call to draw a H of order (n-1)*/  
  
        /* second recursive call to draw a H of order (n-1)*/  
  
        /* third recursive call to draw a H of order (n-1)*/  
  
        /* fourth recursive call to draw a H of order (n-1)*/  
    }  
}
```

Although the above recursive calls can be done in any order, we would like to do them in an order that produces the trees as shown below for $n=3$ (H-tree of order 3). The images below display the interim stages of the H-tree of order-3 creation. Each one should provide a clue as to what order, the 4 trees of order $(n-1)$ are drawn using each tip as the center.



What is the order of 4 recursive calls to draw with proper arguments?

You will receive 3 points for each call if order of calls and arguments are correct. If arguments are correct, but order of calls wrong, the maximum number of points is 6 points.

```
draw(x+d/2, y+d/2)
draw(x+d/2, y-d/2)
draw(x-d/2, y+d/2)
draw(x-d/2, y-d/2)
```

 No files uploaded

Q4 OOP

40 Points

Q4.1

16 Points

Use the given method signatures and instance variables to complete the implementation of the **RaceCar** class.

We require an object to simulate a race car on a track so that we can monitor it's levels and performance, how much fuel it needs at pit stops, etc. A RaceCar object can increase its speed, decrease its speed, increase or decrease its fuel, empty its fuel, and can check the current speed or fuel. The points breakdown will be as follows:

- (4 points) Implement the RaceCar Constructor method with your own parameters.
- (2 points) Increase speed
- (2 points) Decrease speed
- (2 points) Increase Fuel
- (2 points) Decrease Fuel
- (2 points) Empty Fuel
- (1 point) Get Speed
- (1 point) Get Fuel

Begin by initializing the race car such that the current speed is 0.

```

1 public class RaceCar{
2     private int currentFuel; //in gallons
3     private int currentSpeed; //in mph
4     private int speedLimit; // the max speed limit the car can go
5     private int maxFuelCapacity; // the maximum number of gallons the car tank can hold
6     public RaceCar(){
7         //implement this constructor method with your own input parameters
8     }
9     public boolean increaseSpeed(int speedFactor){
10        //implement this method that will increase the speed and
11        //return true if speed was successfully increased, false otherwise
12    }
13    public boolean decrease(int speedFactor){
14        //implement this method. Return true if speed was
15        //successfully decreased, false otherwise
16    }
17    public boolean increaseFuel(int fuel){
18        //implement this method. Return true if fuel was successfully
19        //added, false otherwise
20    }
21    public boolean decreaseFuel(int fuel){
22        //implement this method. Return true if fuel was successfully
23        //decreased, false otherwise
24    }
25    public void emptyFuel(){
26        //implement this method to deplete the tank of all fuel
27    }
28    public int getSpeed(){
29        //return the current speed of the race car
30    }
31    public int getFuel(){
32        //return the current amount of fuel in the race car
33    }
34 }
35

```

```

public class RaceCar{
    private int currentFuel;
    private int currentSpeed;
    private int speedLimit;
    private int maxFuelCapacity;

    public RaceCar(int currentFuel, int speedLimit, int maxFuelCapacity){
        this.currentFuel = currentFuel;
        this.speedLimit = speedLimit;
        this.maxFuelCapacity = maxFuelCapacity;
        currentSpeed = 0;
    }

    public boolean increaseSpeed(int speedFactor){
        if(currentSpeed + speedFactor < speedLimit) {

```

```
currentSpeed += speedFactor;
return true;
}
return false;
}

public boolean decrease(int speedFactor){
if(currentSpeed - speedFactor >= 0) {
currentSpeed -= speedFactor;
return true;
}
return false;
}

public boolean increaseFuel(int fuel){
if(currentFuel + fuel < maxFuelCapacity) {
currentFuel += fuel;
return true;
}
return false;
}

public boolean decreaseFuel(int fuel){
if(currentFuel - fuel >= 0){
currentFuel -= fuel;
return true;
}
return false;
}

public void emptyFuel(){
currentFuel = 0;
}


public int getSpeed(){
return currentSpeed;
}

public int getFuel(){
```



```
return currentFuel:
```

```
}
```

 No files uploaded

Q4.2

4 Points

Which of the following declarations will compile without error?

```
public class BandMember
{
    private String name;
    private int age;

    public BandMember()
    { /* Implementation not shown */ }

    public BandMember(String name, int age)
    { /* Implementation not shown */ }

    // no other constructors
}
```

1. BandMember a = new BandMember();
2. BandMember b = new BandMember("Joey", 24);
3. BandMember c = new BandMember("Johnny", "27");
4. BandMmember d = new BandMember(DeeDee, 24);

- ☐ 1 only
- ☐ 1 and 3 only
- ☒ 1 and 2 only
- ☐ 1 and 4 only
- ☐ 3 only

Q4.3

20 Points

Create a class called **myRect** that will be used to create an object that represents a rectangle or square. This class will have methods to calculate area and perimeter of the rectangle. It will contain the following:

- (1 point) Two **instance variables** called width and height.
- (3 points) Three **constructors**:
 - a default with no parameters that sets the rectangle sides to 1.
 - a constructor that takes one integer value and uses it for width and height, creating a square.
 - a constructor that takes width and height as parameters.
- (2 points) **Getters** (accessor methods) for both the width and height instance variables.
- (3 points) A method called **perimeter** that returns the perimeter of the rectangle.
- (3 points) A method called **area** that return the area of the rectangle.
- (3 points) A **toString** method that returns the String representation of the object. For example, a rectangle with a width of 3 and a height of 2 would return "a 3x2 rectangle".
- (5 points) A **main** method to test the class. The main method is supposed to create **three** myRect objects, a default rectangle, a 5x5 square and a 3x4 rectangle. Then the code should print the width of the first object, the area of the second and print the third to test your toString method.

```
public class myRect {  
    private int width;  
    private int height;  
  
    public myRect() {  
        width = 1;  
        height = 1;  
    }  
  
    public myRect(int x) {  
        width = x;  
        height = x;  
    }  
}
```

```
public myRect(int w, int h) {  
    width = w;  
    height = h;  
}  
  
public int getWidth() {  
    return width;  
}  
  
public int getHeight() {  
    return height;  
}  
  
public int perimeter() {  
    return 2*width + 2*height;  
}  
  
public int area() {  
    return width*height;  
}  
  
public String toString() {  
    if(width == height) {  
        return "a " + width + "x" + height + " square";  
    }  
    else {  
        return "a " + width + "x" + height + " rectangle";  
    }  
}  
  
public static void main (String[] args) {  
    myRect object = new myRect();  
    myRect object2 = new myRect(5);  
    myRect object3 = new myRect(3, 4);  
    System.out.println(object.getWidth());  
    System.out.println(object2.area());  
    System.out.println(object3);  
}  
}
```

Q5 Sorting & Searching

20 Points

Q5.1

2.5 Points

If you run the selection sort algorithm on an array of n items, how many times are two array items swapped?

```
public class SelectionSort{  
    public static void selection(int[] a){  
        int n = a.length;  
        for (int i= 0; i < n; i++){  
            int minPos = i;  
            for (int j = i+1; j < n; j++){  
                if (a[j] < a[minPos]){  
                    minPos = j;  
                }  
            }  
            int temp = a[i];  
            a[i] = a[minPos];  
            a[minPos] = temp;  
            //printArray(a);  
        }  
    }  
}
```

n-1

Q5.2

2.5 Points

Suppose that selection sort on an array of 100 items has completed 31 iterations of the outer loop. How many items are now guaranteed to be in their final spot (never to be moved again)?

31

Q5.3

5 Points

Mergesort makes two recursive calls.

```
public class Merge
{
    private static String[] aux;
    public static void merge(String[] a, int lo, int mid, int hi)
    {
        //copy lower half of the array into b
        String[] b = new String[mid-lo+1];
        for(int i=0;i<=mid-lo;i++) { b[i] = a[lo+i]; }
        int i=0,j=mid+1,k=lo;
        while(i <= mid-lo && j <= hi) {
            if (a[j].compareTo(b[i]) < 0) {
                // a[j] < b[i]
                a[k] = a[j]; k+=1; j+=1;
            } else {
                a[k] = b[i]; k+=1; i+=1;
            }
        }

        // copy remaining
        while ( i <= mid-lo ) { a[k] = b[i]; k+=1; i+=1; }
        while ( j <= hi ) { a[k] = a[j]; k+=1; j+=1;}
    }

    public static void sort(String[] a)
    {
        aux = new String[a.length]; // Allocate just once!
        sort(a, 0, a.length);
    }

    public static void sort (String[] a, int lo, int hi) {
        if (lo >= hi) return;
        int middle = (lo+hi)/2;
        sort(a, lo, middle);
        sort(a, middle+1, hi);
        merge(a, lo, middle, hi);
    }
}
```

A. **(2.5 points)** What can be said about the array after these recursive calls finish, but before the merge step?

Both halves of the array are sorted but only within themselves. The array is not fully sorted as a whole.

B. **(2.5 points)** Show the content of the following array `[9, 16, 5, 2, 17, 4, 1]` after the two recursive calls finish, but before the merge step.

`[2, 5, 9, 16], [1, 4, 17]`

Q5.4

5 Points

Suppose that we are sorting an array of eight integers using a quadratic sorting algorithm. After four iterations of the algorithm's outer loop, the array elements are ordered as shown here: `[2 4 5 7 8 1 3 6]`.

Which statement is correct? Note that our selection sort picks largest items first.

- ☐ The algorithm might be either selection sort or insertion sort.
- ☐ The algorithm might be selection sort, but it is not insertion sort.
- ☒ The algorithm is not selection sort, but it might be insertion sort.
- ☐ The algorithm is neither selection sort nor insertion sort.

Q5.5**2.5 Points**

You are tasked with designing an application that will search an array of 100 items **ONCE**. Assume that you want the application to have the fastest possible running time, which of the following options would you choose?

- ☐ Sort the array using mergesort and then search using binary search.
- ☐ Sort the array using insertion sort and then search using binary search.
- ☒ Sort the array using selection sort and then search using binary search.
- ☐ Search the array using sequential search.

Q5.6**2.5 Points**

You are tasked with designing an application that will search a large array a huge number of times. Assume that you want the application to have the fastest possible running time, which of the following options would you choose?

- ☒ Sort the array using mergesort and then search using binary search.
- ☐ Sort the array using insertion sort and then search using binary search.
- ☐ Sort using selection sort and then search using binary search.
- ☐ Search the array using sequential search.

Q6 Complexity

20 Points

Q6.1

10 Points

For the following ten algorithms, select their Big O Complexity. Select the most efficient Big O (the fastest) for the worse case scenario of the data. Example: If I have an algorithm that adds 1 to every item in an integer array the Big O is $O(n)$. $O(n \log n)$ also works, but it's not needed. $O(n)$ is the fastest.

For each problem choose from: $O(1)$, $O(n)$, $O(n^2)$, $O(n^3)$, $O(n!)$ or Not Applicable (N/A).

1. **(1 point)** Find the middle element in an array

- ☒ $O(1)$
- ☐ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☐ Not Applicable N/A

2. **(1 point)** Determine if an unsorted array has any duplicates

- ☐ $O(1)$
- ☒ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☐ Not Applicable N/A

3. **(1 point)** Determine if a given array is sorted

- ☐ $O(1)$
- ☒ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☐ Not Applicable N/A

4. **(1 point)** Binary search for a target value in an unsorted array

- ☐ $O(1)$
- ☐ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☒ Not Applicable N/A

5. **(1 point)** Convert an array of size n in descending order to an array in ascending order.

That is, `[3 2 1] -> [1 2 3]`

- ☐ $O(1)$
- ☒ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☐ Not Applicable N/A

6. **(1 point)** Given a $n \times n$ 2D array of integers, check if there are two elements on the diagonal whose sum is 0. The diagonal elements in a 2D array are given by `a[i][i]` (uses the same index for row and column)

- ☐ $O(1)$
- ☐ $O(n)$
- ☒ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☐ Not Applicable N/A

7. **(1 point)** Finding all permutations of the n elements in the array A

- ☐ $O(1)$
- ☐ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☒ $O(n!)$
- ☐ Not Applicable N/A

8. (1 point) Sort an array of size n using selection sort

- ☐ $O(1)$
- ☐ $O(n)$
- ☒ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☐ Not Applicable N/A

9. (1 point) Find the second largest element in an unsorted array

- ☐ $O(1)$
- ☒ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☐ Not Applicable N/A


10. (1 point) Copy the values from a 2d array to another 2d array (assume they are the same size).

- ☐ $O(1)$
- ☒ $O(n)$
- ☐ $O(n^2)$
- ☐ $O(n^3)$
- ☐ $O(n!)$
- ☐ Not Applicable N/A

Q6.2**5 Points**

The Big O Complexity of a Mergesort is $O(n \log n)$. Explain why.

It is $O(n \log n)$ because Mergesort split an array into halves until it cannot be split any further.

 No files uploaded

Q6.3**5 Points**

If I have an algorithm that runs at $O(n^2)$ how much more time does it take to run my algorithm if I have 4 times as much data as the initial run?

16 times longer than the initial run.

 No files uploaded