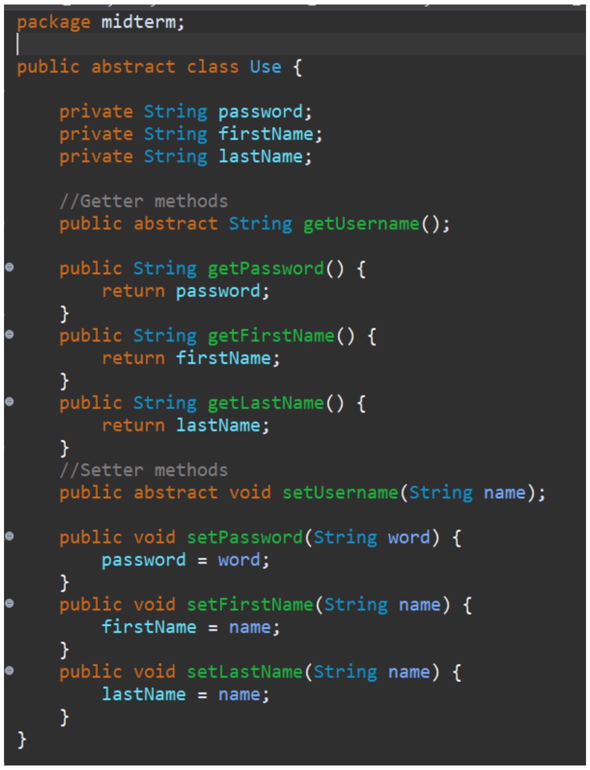
Conceptual Questions (50pts – 5pts/question)

Submission Instruction:
Write the answers of these questions in a pdf as **fullname_conceptquestions.pdf**
If you write code pls add it to the pdf file (do not submit java code in .java files for these questions)

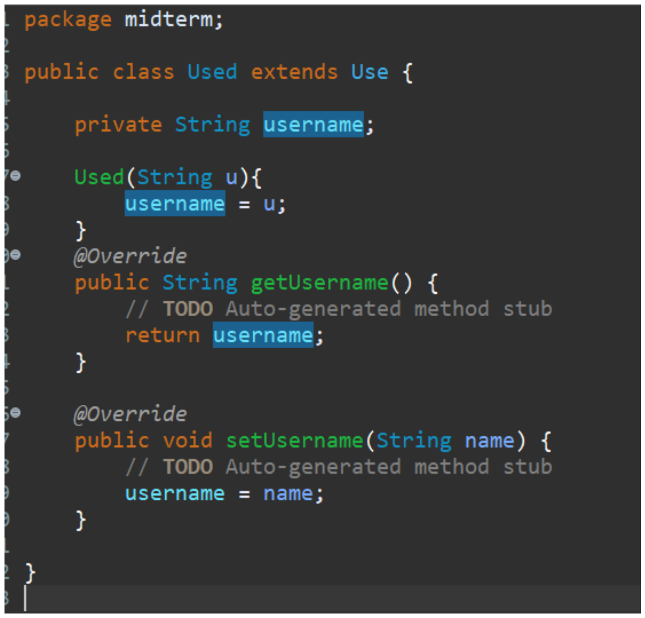
1. Explain the difference between an abstract class and an interface.
Generally Abstract classes are more flexible than Interfaces. Abstract Classes can have method bodies while Interfaces can only have method signatures.

2. How is the keyword super being used in the context of OOP?
In OOP, the keyword super is used to call methods or other elements from parent classes aka using inheritance.

3. Provide code in Java that demonstrate the use of an Abstract class (your abstract class should contain at least one abstract method and there is at least one class that inherits from the abstract class you defined)



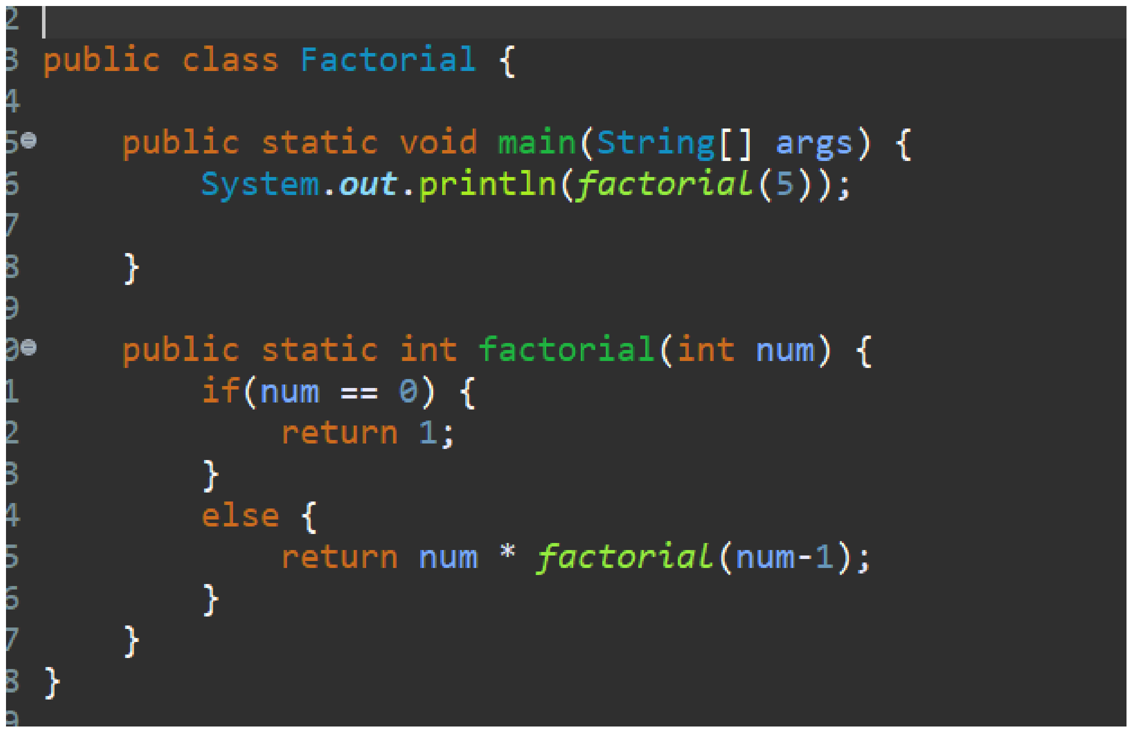
```java
package midterm;

public abstract class Use {

    private String password;
    private String firstName;
    private String lastName;

    //Getter methods
    public abstract String getUsername();

    public String getPassword() {
        return password;
    }
    public String getFirstName() {
        return firstName;
    }
    public String getLastName() {
        return lastName;
    }
    //Setter methods
    public abstract void setUsername(String name);

    public void setPassword(String word) {
        password = word;
    }
    public void setFirstName(String name) {
        firstName = name;
    }
    public void setLastName(String name) {
        lastName = name;
    }
}
```



```java
package midterm;

public class Used extends Use {

    private String username;

    Used(String u){
        username = u;
    }
    @Override
    public String getUsername() {
        // TODO Auto-generated method stub
        return username;
    }

    @Override
    public void setUsername(String name) {
        // TODO Auto-generated method stub
        username = name;
    }

}
```

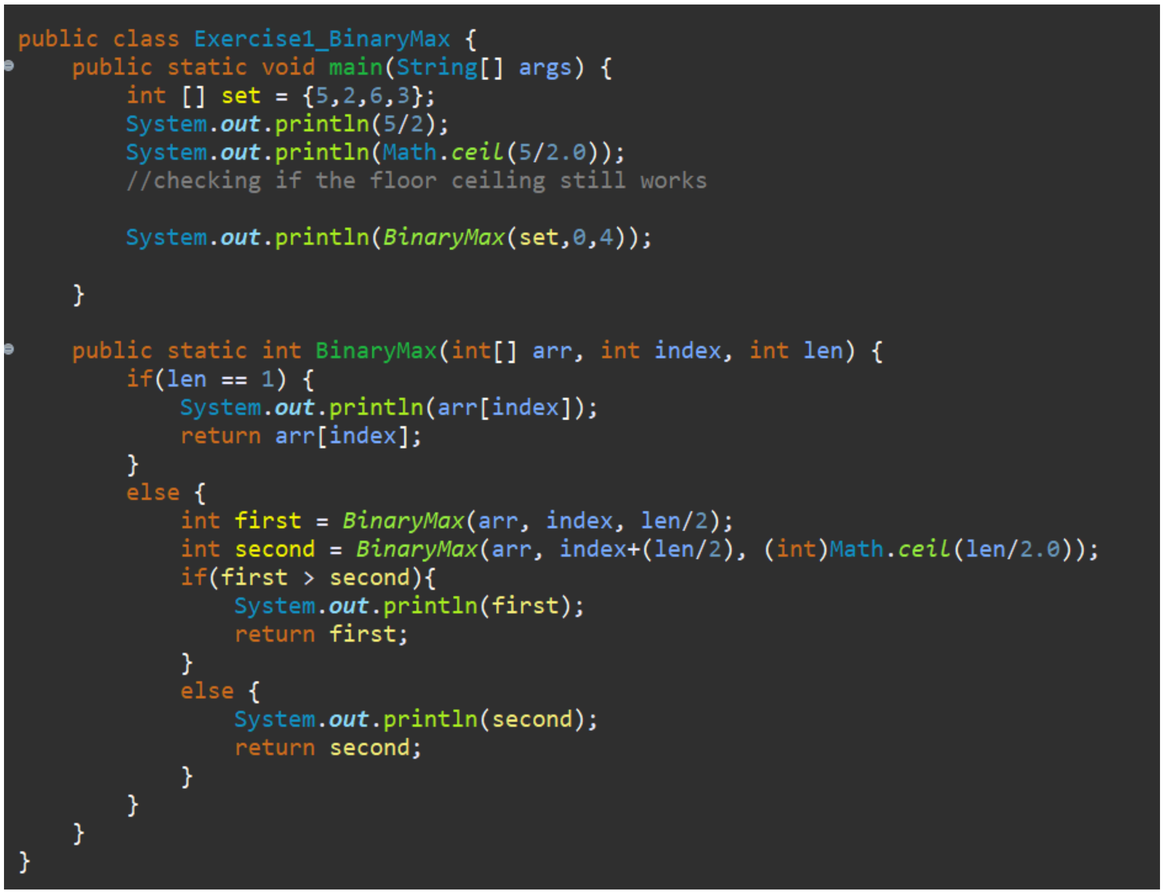
4. What is the difference between binary recursion and linear recursion? Provide examples.
The difference between binary and linear recursion is that linear recursion calls itself only once within the method body, while binary recursion calls itself Twice in its own method body.

Linear Recursion Example:



```java

public class Factorial {

    public static void main(String[] args) {
        System.out.println(factorial(5));

    }

    public static int factorial(int num) {
        if(num == 0) {
            return 1;
        }
        else {
            return num * factorial(num-1);
        }
    }
}

```

Binary Recursion Example:



```java
public class Exercise1_BinaryMax {
    public static void main(String[] args) {
        int [] set = {5,2,6,3};
        System.out.println(5/2);
        System.out.println(Math.ceil(5/2.0));
        //checking if the floor ceiling still works

        System.out.println(BinaryMax(set,0,4));

    }

    public static int BinaryMax(int[] arr, int index, int len) {
        if(len == 1) {
            System.out.println(arr[index]);
            return arr[index];
        }
        else {
            int first = BinaryMax(arr, index, len/2);
            int second = BinaryMax(arr, index+(len/2), (int)Math.ceil(len/2.0));
            if(first > second){
                System.out.println(first);
                return first;
            }
            else {
                System.out.println(second);
                return second;
            }
        }
    }
}
```

5. What is an activation record? Which data structure supports a recursive class?
An activation record is a record of Instances of recursive methods being called within itself. It includes which instances are being put on hold in favor of the more recent instance being activated.

An Activation record is supported by the Stack Data Structure.

Where the first thing put on the stack is also the last thing you get to taking care of as it's buried under the newest instance being called at the top.

6. Is a recursive solution more efficient than an iterative solution?
Not necessarily. Sometimes recursion may take more operations than an iterative solution. Also the Stack data structure, putting methods on pause in favor of taking care of it's newer instance of itself, takes time and space that the problem might not need in order to solve it.

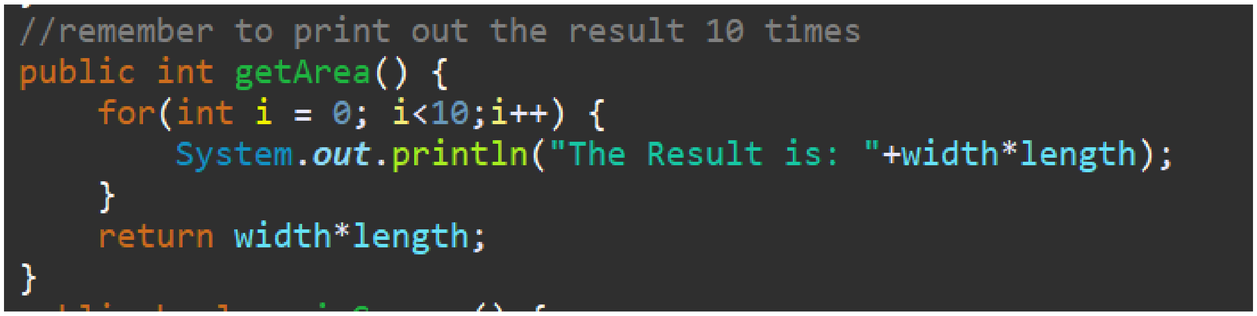
7. We discussed in class two main way of analyzing running time of algorithms? Explain both approaches.
Experimental Analysis and Theoretical Analysis

Experimental Analysis involves actually running the programs and measuring the time it takes to run on a specific machine. Experimental Analysis may be flawed in that the hardware and software of the machine may affect the runtime, alongside any potential lag in human or machine measuring the time itself.

Theoretical Analysis involves taking a look at the algorithm's code itself. Usually by imagining it taking the worst possible input, one that would maximize the amount of work to run through. It's in theoretical analysis that we use Big-O notation like O(n) or O(1) or O(log(n))

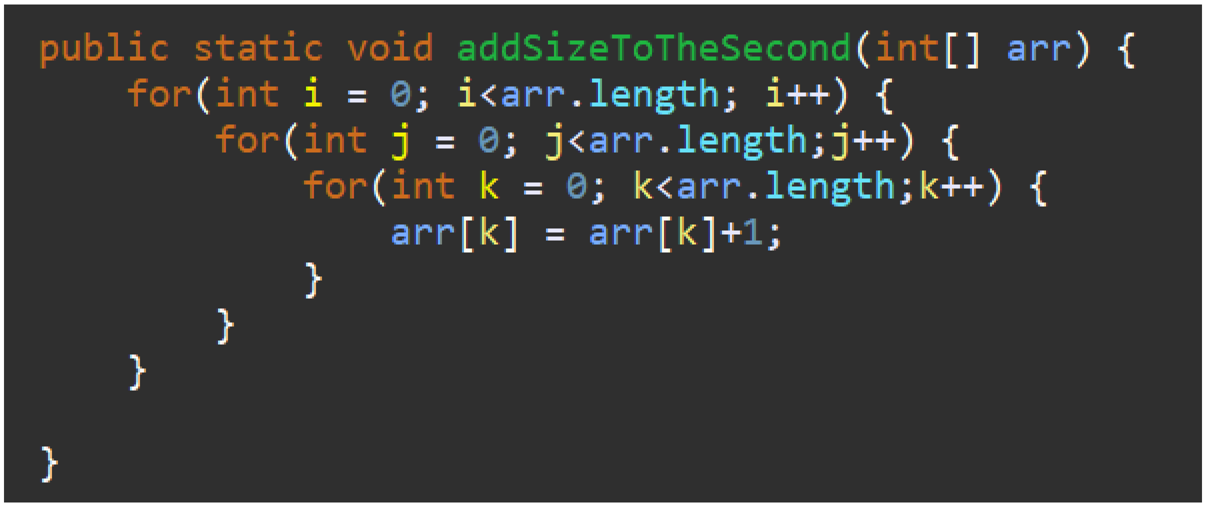
8. Provide examples of algorithms that are of the following running times: O(1), O(N^3), O(n Log(n)) (one algorithm for each running time class)

O(1) example:



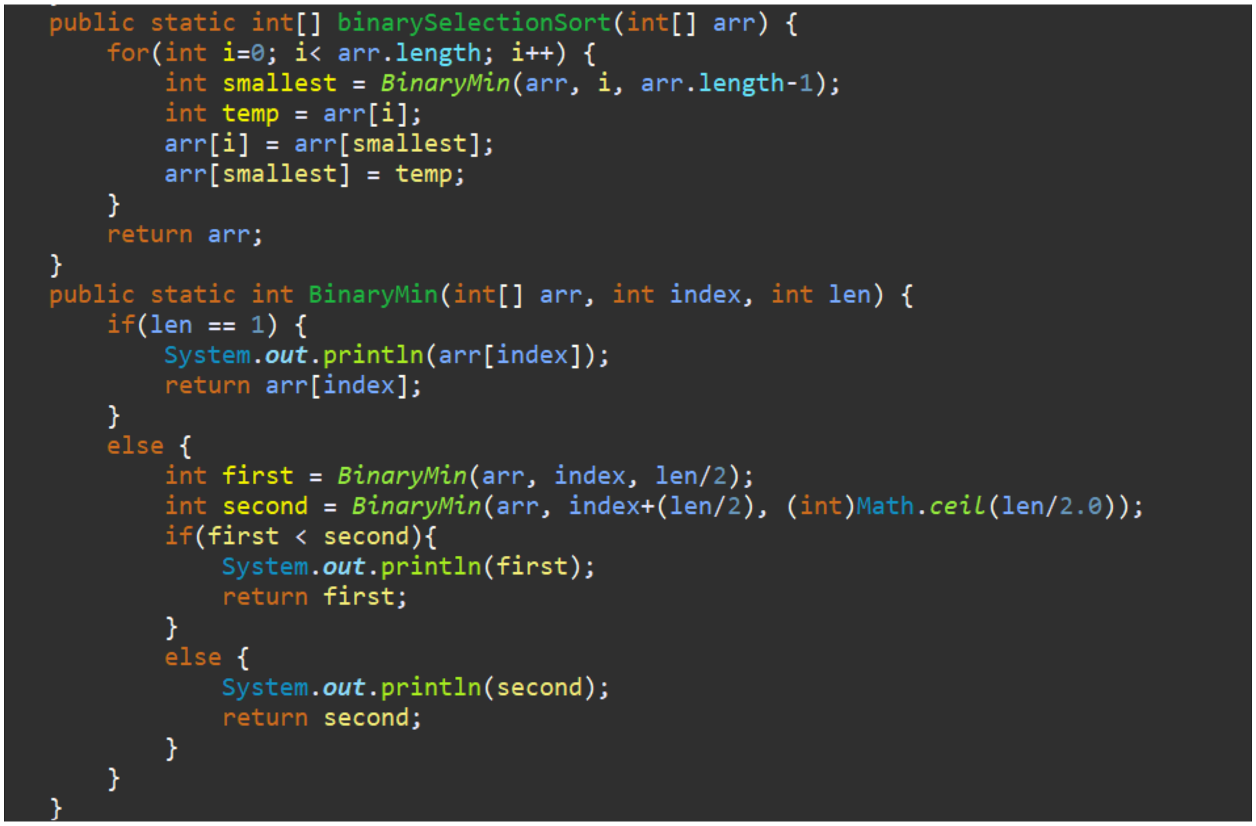
```
//remember to print out the result 10 times
public int getArea() {
    for(int i = 0; i<10;i++) {
        System.out.println("The Result is: "+width*length);
    }
    return width*length;
}
```

O(N^3) example:



```java
public static void addSizeToTheSecond(int[] arr) {
    for(int i = 0; i<arr.length; i++) {
        for(int j = 0; j<arr.length;j++) {
            for(int k = 0; k<arr.length;k++) {
                arr[k] = arr[k]+1;
            }
        }
    }

}
```

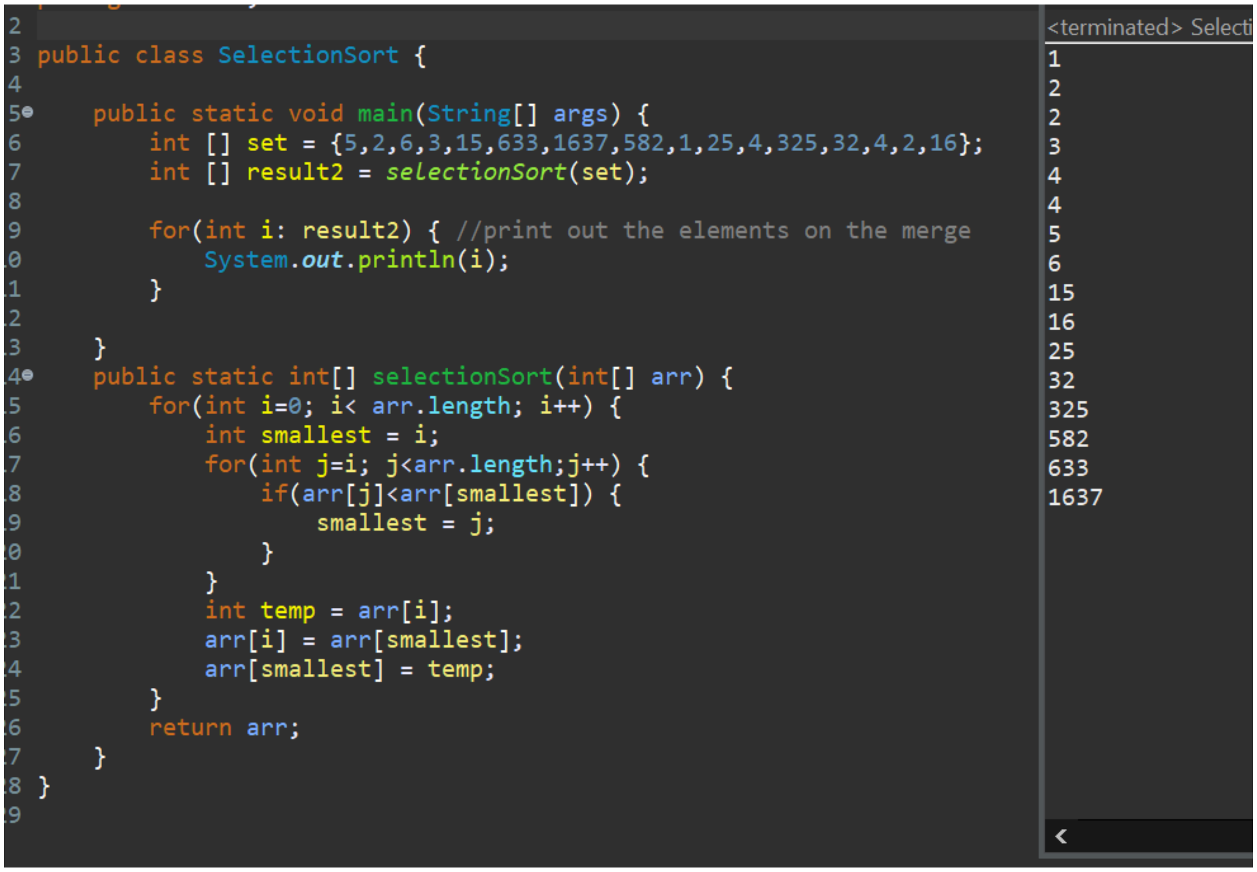
O(nLog(n)) example:



```java
public static int[] binarySelectionSort(int[] arr) {
    for(int i=0; i< arr.length; i++) {
        int smallest = BinaryMin(arr, i, arr.length-1);
        int temp = arr[i];
        arr[i] = arr[smallest];
        arr[smallest] = temp;
    }
    return arr;
}
public static int BinaryMin(int[] arr, int index, int len) {
    if(len == 1) {
        System.out.println(arr[index]);
        return arr[index];
    }
    else {
        int first = BinaryMin(arr, index, len/2);
        int second = BinaryMin(arr, index+(len/2), (int)Math.ceil(len/2.0));
        if(first < second){
            System.out.println(first);
            return first;
        }
        else {
            System.out.println(second);
            return second;
        }
    }
}
```

9. Provide the idea behind Selection Sort, and provide full implementation of it, and explain it is running time.

Selection Sort is based on going through the array looking for the smallest number in it and pushing it to the beginning of the array and repeating that process until everything is in ascending order.



```java

public class SelectionSort {

    public static void main(String[] args) {
        int [] set = {5,2,6,3,15,633,1637,582,1,25,4,325,32,4,2,16};
        int [] result2 = selectionSort(set);

        for(int i: result2) { //print out the elements on the merge
            System.out.println(i);
        }

    }
    public static int[] selectionSort(int[] arr) {
        for(int i=0; i< arr.length; i++) {
            int smallest = i;
            for(int j=i; j<arr.length;j++) {
                if(arr[j]<arr[smallest]) {
                    smallest = j;
                }
            }
            int temp = arr[i];
            arr[i] = arr[smallest];
            arr[smallest] = temp;
        }
        return arr;
    }
}
```

```
<terminated> Selecti
1
2
2
3
4
4
5
6
15
16
25
32
325
582
633
1637
```

10. Explain the concept of serialization. What were the advantages of using serialization in HW1.

The main idea of serialization is to store files and encode them into a byte stream. This byte stream is a form that doesn't exclude other platforms from being able to decode and access those files. Ultimately the advantage of using serialization is it's multi-platform access.