1. Explain the difference between an abstract class and an interface.
Abstract classes use abstract keyword and extend keyword since it can be “extended”, while interface uses interface implements to be “implemented”.
Abstract classes can have both abstract and non-abstract methods, while interfaces can only have abstract methods; Abstract classes can not have multiple inheritances, while interfaces can. Abstract classes can implement interfaces, while interfaces can not implement abstract classes. Interfaces are fully abstract while abstract classes are only partly abstract.

2. How is the keyword super being used in the context of OOP?
Super keyword in oop is used to refer parent class objects, mainly used to refer the immediate parent class instance variable, method, and constructors. It is an important concept of 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)

```
public abstract class Animals {

        public abstract void name();

}
public class Dog extends Animals
{
        public void name()
        {
                System.out.println("dog");
        }
}
public class Cat extends Animals
{
        public void name()
        {
                System.out.println("cat");
        }
}
```

Here, the child classes Dog and Cat extend the abstract parent class Animal with an abstract method name(). They both have their own name() method.

4. What is the difference between binary recursion and linear recursion? Provide **examples**.

Linear recursion performs a single recursive call, and during the call, it chooses one of them to perform, and it comes down to the base case. An example of a linear recursive method is factorial

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

Binary recursion is when two recursive calls for each non-base case. An example of a binary recursive method is binarySum.

```
public static int BinarySearch(int key, int[] A, int LI, int HI)
        {
                int mid = (LI+HI)/2;
                if(LI>HI)
                {
                        return -1;
                }

                if(key==A[mid])
                {
                        return mid;
                }
                else if(key<A[mid])
                {
                        return BinarySearch(key, A, LI, mid-1);
                }
                else
                {
                        return BinarySearch(key, A, mid+1, HI);
                }
```

```
	}
```

5. What is an activation record? Which data structure support recursive calls?
Activation record is a private block of memory associated with an invocation of a procedure, used to manage information needed by a single execution. It is created on stack, and it contains the return address and the current status of the caller. After the recursive method call finished, the activation record popped the stack and the suspended parts continues.

6. Is a recursive solution more efficient than an iterative solution?
Usually, iterative solutions are more efficient than recursive solutions. Recursion exhausts memory resources through the call of stack.

7. We discussed in class two main way of analyzing running time of algorithms? Explain both approaches.
There are two main ways to analyze the running time of an algorithm, one is experimental and the other one is theoretical. The experimental method uses different outputs and records the time it took to finish the task. The theoretical method analyzes the algorithms without implementing them or using any hardware and operating systems environment. Big O notation is used for this, and it calculates the runtime of the algorithm in terms of the function with the input of N. Theoretical approach is much more efficient and explanatory when it comes to analyzing the running time of algorithms.

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)
```
public String o1(String a)
	{
		return a;
	}
```
O(N^3)
```
public void o3(int n)
	{
		for (int i = 1; i <= n; i++) {
		   for(int j = 1; j <= n; j++) {
```

```
                    for(int k=0; k<=n; k++)
                    {
                      System.out.println(k);
                    }
                  }
                }
        }
```

O(nlog(n)): the mergeSort algorithm we wrote is O(nlog(n)) time compexity

```
public static int[] mergeSort(int[] a, int[] b)
        {
                int size = a.length+b.length;
                int[] c = new int[size];
                int an = 0;
                int bn = 0;
                for(int i=0; i<size; i++)
                {
                        if(an<a.length&&bn<b.length)
                        {

                                if(a[an]<b[bn])
                                {
                                        c[i]=a[an];
                                        an++;
                                }
                                else if(a[an]>b[bn])
                                {
                                        c[i]=b[bn];
                                        bn++;
                                }
                                else
                                {
                                        c[i]=a[an];
                                        c[i+1]=b[bn];
                                        i++;
                                        an++;
                                        bn++;
                                }
                        }
                        else if(an==a.length && bn<b.length)
                        {
                                c[i]=b[bn];
                                bn++;
                        }
                        else if(bn==b.length && an<a.length)
                        {
                                c[i]=a[an];
                                an++;
                        }
                }
                return c;
        }
```

9. Provide the idea behind Selection Sort, and provide full implementation of it, and explain it is running time.
Selection sort is an algorithm that sorts an array by finding the minimum element and placing it at the beginning of the array. Every time the selection sort is called, the minimum element of the subarray will be converted to the front, and therefore a complete iteration will sort the unsorted array in ascending order.

```
public class selectionSort {
        public void selectionSort(int a[])
        {
                int l = a.length;
                int min;
                for(int i=0; i<l-1; i++)
                {
                        min = i;
                        for(int j=i+1; j<l; j++)
                        {
                                if(a[min]>a[j])
                                {
                                        min=j;
                                }
                        }
                        int temp = a[min];
                        a[min] = a[i];
                        a[i] = temp;
                }
```

```
    }
}
```
The running time of the selection sort method is O(N^2). There are two iterations in this method, which gives us two O(N) time complexity methods. The result of these two nested for loops will be the product of them, which is O(N^2).

10. Explain the concept of serialization. What were the advantages of using serialization in HW1
Serialization is a way of organizing objects and converting objects into a byte stream, which can also be converted back. We need to use serialization with java.io.Serializable interface implemented to any superclasses. It is a convenient way for the computer to read the files efficiently. Using serialization in our university course selection system largely decreases the memory required when accessing the courses and the students lists.