

1. Matching [8 pts]

For each algorithm name, select the correct textual description of the algorithm.

1. _____ Selection Sort
2. _____ Linear Search
3. _____ Quick Sort
4. _____ Binary Search
 - A. Adjacent items in the array are compared to each other and swapped as needed until the array is sorted.
 - B. Selects a pivot element, groups the data relative to the pivot, recurses on each subpart.
 - C. On a sorted array, an element is found quickly by eliminating half of the array with each comparison.
 - D. The Y combinator is applied to each element in the array until a least upper bound can be established resulting in a sorted array.
 - E. The array is scanned sequentially until a desired element is found.
 - F. The maximum (or minimum) element of the unsorted portion of the array is found, then moved to its final position. This process is repeated until the array is sorted.
 - G. None of the above.
 - H. All of the above.

2. Tracing [16 pts]

What is the output when the main method in the code below is run?

```
public class Trace1{
    public static void mystery1(int x, int[] data){
        if(x==0){
            System.out.println("All done");
            return;

        }
        data[x]++;
        x--;
        mystery1(x,data);
        System.out.println("x= "+x);
    }
    public static void main(String[] args){
        int[] data={0,1,2,3};
        int x=2;
        mystery1(x,data);
        System.out.println("x is now " +x);
        for(int i=0;i<data.length;i++){
            System.out.println(data[i]);
        }
    }
}
```

3. Polymorphism [12 pts]

- 4 (a) Explain the concept of polymorphism. Your explanation should make specific reference to
- Inheritance
 - Abstract Classes
 - Interfaces
- 4 (b) Give an example of what polymorphism allows a programmer to do. Be sure to mention
- Collections (arrays, Vectors, etc).
 - Methods
- (you do not have to write code for this problem, but if short code fragments help your explanation, you may include them)
- 4 (c) Explain when/why casting is needed with respect to Objects.

4. Dynamic Binding [12 pts]

Assume that you are given the following classes:

- **Animal** is abstract, and has the abstract method **public int foodNeeded()**, a method which specifies how many pounds of food the animal needs.
- **Cat** extends **Animal** and implements **foodNeeded()** so that it returns 3.
- **Lion** extends **Cat** and implements **foodNeeded()** so that it returns 50.
- **Elephant** extends **Animal** and implements **foodNeeded()** so that it returns 147.

8

- (a) Write the method **public int totalFoodNeeded(Animal[] myZoo)** which computes the total food needed for all **Animals** in **myZoo**. This method should be useable with other **Animal** subclasses that may be written later. For this method, you may not use

1. The **instanceof** operator
2. Casting of any sort
3. The **getClass()** method

If you violate the above restriction, you will receive no credit for this question.

Given the following **Animal[]**:

```
Animal[] testZoo={new Cat(), new Lion(), new Lion(), new Elephant()};
```

the **totalFoodNeeded** method should return 250.

```
public int totalFoodNeeded(Animal[] myZoo) {
```

```
}
```

4

- (b) Explain how your method accomplishes its task, with specific reference to the concept of Dynamic Binding.

5. **Arrays [16 pts]**

Write the method **public Vector arrayToVect(int[][] data)** which creates a Vector with all of the elements of **data** in it. The elements from the array may appear in the Vector in any order, as long as the Vector contains the same items as the array. Remember that the Integer class may come in handy for this problem. You may not assume that **data** will be rectangular, it may be "jagged". For example, given the array

```
{{1,2,3,3},  
{4,5}}
```

The answer Vector could be [1,2,3,3,4,5].

```
public Vector arrayToVect(int[][] data) {
```

```
}
```

6. **Sorting [16 pts]**

Assume that you have the following correctly working method already implemented:

- `public Comparable[] merge(Comparable[] a1, Comparable[] a2)`

Write the method **`public void mergeSort(Comparable[] data)`** which sorts the array passed in into ascending order. The **`merge`** method given will merge the two arrays given in ascending order.

```
public void mergeSort(Comparable[] data) {
```

```
}
```

7. Recursion [20 pts]

A sound processor filters out frequencies below a low frequency threshold and above a high frequency threshold. Such a processor might be used as part of an MP3 encoder to reduce the amount of data to be processed. Given that a frequency is a number, develop the method, **filterFreq**, which consumes two threshold frequencies (low and high) and an array of frequencies and returns an array of those frequencies between and including the thresholds. For this problem, you may assume that you have the method:

```
public static int[] arrayRest(int[] data)
```

which returns an array that has all elements of data except the first.

- 5 (a) As a first step, write a helper method called `public static int[] arrayCons(int x, int[] data)`. This method should return an array containing x as its first element, followed by all of the elements of data. For example,

```
arrayCons(5, {4,3,2}) would be {5,4,3,2}
```

You do not have to use recursion for this sub-part of the problem (although, you may if you find it useful)

```
public int[] arrayCons(int x, int[] data) {
```

```
}
```

(Question continues on next page ⇒)

- 15 (b) Now write the **filterFreq** method. You **must use recursion only** for this problem- if you do not use recursion, or you use any iteration, you will receive no credit. You may assume that the **arrayCons** method that you just wrote works as described for this part of the problem, and may use it in your answer. Some examples of how this method works include

```
int freqs[] = { 300, 350, 200, 600, 20 };
filterFreq( 250, 800, freqs ) should return { 300, 350, 600 }
filterFreq( 250, 500, freqs ) should return { 300 , 350 }
filterFreq( 250, 280, freqs ) should return { }
```

```
public int[] filterFreq(int lowBound, int highBound, int[] data) {
```

```
}
```