CST8130: Data Structures
Midterm Test #1 : Version B

Name: _____Solutions__________________________

Instructions

Answer all questions on the test paper. There are 17 questions worth a total of 32 marks.
If you have any questions, raise your hand and I will come and answer.

Part 1 – Multiple Choice (14 marks) Answer these questions in Blackboard.

The following class will be used for the next group of questions:

```
public class Student {

     private  int studentNumber;
     protected String studentName;
     private float studentGPA;

     public Student (int studentNumber, String studentName, float studentGPA) {
               this.studentNumber = studentNumber;
               this.studentName = studentName;
               this.studentGPA = studentGPA;
     }

….other methods………..

     public String toString( )  {
            return studentName + “ “ + studentNumber + “ has average “ + studentGPA;
     }

     public boolean  isEligible() {
            if (studentGPA >= 2.7f)
                     return true;
            else return false;

    }

public class CoopStudent extends Student {
    private String employerName;

    public CoopStudent(int studentNumber, String studentName, float studentGPA,
                                                                        String employerName) {
                 // write code here for question which follows
     }

…..  other methods…………..

    public String toString() {
              return super.toString() + “ worked for employer “ + employerName;
    }
}
```

1. What would be the *correct* code to initialize all data members in the initial constructor of **CoopStudent** class?
   - **a) super();**
     this.employerName = employerName;
   - **b) this.studentNumber = studentNumber;**
     this.studentName = studentName;
     this.studentGPA = studentGPA;
     this.employerName = employer;
   - **c) super (studentNumber, studentName, studentGPA);**
     this.employerName = employerName;
   - d) a or b
   - e) b or c
   - f) a or b or c
   - g) none of the above.

2. Given the declaration **Student student = new CoopStudent (1234, "Linda", 3.4f, "Algonquin");** what will display from the statement**?**

```
if (student.isEligible() )
        System.out.print (student + " is eligible");
else  System.out.print (student + "is not eligible");
```

   - a) student is eligible
   - b) student is not eligible
   - c) Linda 1234 has average 3.4 worked for Algonquin is eligible
   - d) Linda 1234 has average 3.4 worked for Algonquin is not eligible
   - e) None of the above (allowed due to typo)

3. Given the declaration **CoopStudent coopStudent = new CoopStudent (1234, "Linda", 3.4f, "Algonquin");** what will display from the statement**?**

```
if (coopStudent.isEligible() )
        System.out.print (coopStudent + " is eligible");
else  System.out.print (coopStudent + "is not eligible");
```

   - a) coopStudent is eligible
   - b) coopStudent is not eligible
   - c) Linda 1234 has average 3.4 worked for Algonquin is eligible
   - d) Linda 1234 has average 3.4 worked for Algonquin is not eligible
   - e) None of the above(allowed due to typo)

4. If a reference, an int and a float all use 1 Byte of memory and a String uses 2 Bytes, how many Bytes of memory does the following declaration have in memory? **Student [ ] students = new Student[5];**

   a) 5 b) 6 c) 25 d) 26 e) 35 f) 36

5. If a reference, an int and a float all use 1 Byte of memory and a String uses 2 Bytes, how many Bytes of memory does the following declaration have in memory? **Student [ ] students = new CoopStudent[5];**

   b) 5 b) 6 c) 25 d) 26 e) 35 f) 36

6. What is the Big-O for the following algorithm? Assume that `DoIt (…)` has an efficiency of O(*n*).

```
   j = n;
   while (j > 0) {
     DoIt(...);
     j = j / 2;
     }
   }
  i = 1;
  while (i < n){
     i = i * 2;
     j = n;
     while (j > 0) {
        DoIt(...);
         j--;
     }
   }
```

What is the Big-O for the above algorithm? Assume that `DoIt(...)` has an efficiency of $O(n)$.

a) $O(n \log_2 n)$ b) $O(n^3)$ c) $O(n^2 \log_2 n)$ d) $O(\log_2 n^2)$ e) $O(n^2)$

7. Which of the following is the best algorithm measurement?

a) $O(n \log_2 n)$ b) $O(2)$ c) $O(n^2 \log_2 n)$ d) $O(\log_2 n^2)$ e) $O(n^2)$

8. Which of the following is the worst algorithm measurement?

b) $O(n \log_2 n)$ b) $O(2)$ c) $O(n^2 \log_2 n)$ d) $O(\log_2 n^2)$ e) $O(n^2)$

9. Given the following code, what is output?

```
public static int recurse (int x) {
      if (x < 1)
         return x;
     else
       return (x + recurse (x-2));
}

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

a) 9 b) 8 c) 5 d) 12 e) 10 f) none of the other answers

10. Given the following code, what is output?

```
public static int recurse (int x) {
      if (x < 1)
         return x;
     else
       return (x + recurse (x-2));
}

public static void main(String [] args)  {
         System.out.println (recurse(6));
}
```

a) 9 b) 8 c) 5 d) 12 e) 10 f) none of the other answers

11. Given the following code, what is output? Note that the arguments ***x*** and ***n*** are exchanged in the recursive calls.

```
public static int recurse2(int n, int x) {
    if (x < 1)
      return 1;
    else
      return n + recurse2(x-1, n-1);
}


public static void main(String [] args)  {
        System.out.println ( recurse2(4, 3));
}
```

a) [✓ 9] b) 8 c) 0 d) 12 e) none of the above

12. Given the following code, what is output?

```
public static int doIt (int num) {
            int total = 0;
            for (int i=0; i<num; i++)
                        for (int j=0; j<num-i-1; j++)
                                    total++;
           return total;
}


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

a) 9 b) 25 c) 14 d) 15 e) [✓ 10] f) none of the other answers

13. What is the measurement of a sequential insertion of the xth object to a dynamically allocated array of n items when x < n?

a) $O(n \log_2 n)$ [✓ b) $O(1)$] c) $O(n^2 \log_2 n)$ d) $O(\log_2 n^2)$ e) $O(n^2)$

14. What is the measurement of a sequential search for an object in an unsorted dynamically allocated array of n items?

a) $O(n \log_2 n)$ b) $O(1)$ c) [✓ $O(n)$] d) $O(\log_2 n^2)$ e) $O(n^2)$

Part 2 – Short Answer (8 marks)

15. Two of the sorting algorithms we studied had O($n \log_2 n$) efficiency (MergeSort and QuickSort). If we have a list of objects that contain large amounts of data, what property of these algorithms might cause us to choose QuickSort over MergeSort? ***(2 marks)***

Merge Sort uses a complete copy of the array of references…..QuickSort does not

16. Using the class **Student** and **CoopStudent** from above, assume that each of the classes has a method - **public boolean addKeyboard() -** which prompts the user to input the data for the respective class, and

```
 public class Students {
        private int numStudents = 0;
        private int maxStudents = 10;
        private Student [ ] students = new Student[maxStudents];

        public boolean addStudent(int type) {
                        // write code that goes here
        }

}
```

Write the code for method **addStudent** which will use the parameter **type** (1 for regular student, 2 for coop student), add the appropriate student into the array using method **addKeyboard** to get the data for the student being added. No exception handling is required. ***(6 marks)***

```
public boolean addStudent (int type) {
    if (numStudents >= maxStudents)
            return false;

    if (type == 1) {
        students[numStudents] = new Student();
    else
        students[numStudents] = new CoopStudent();

   return students[numStudents++].addKeyboard();
}
```

Part 3 – Programming Question (10 marks)

13. We studied sorting algorithms because with a sorted list of data, we can use binary search (O(logn)) to then find a particular instance of data. The binary search algorithm in general looks in the middle of the list first – if that element is not the one being searched for, it repeats the process using the top half or bottom half of the list since the list is sorted. This is an iterative algorithm, not recursive. Using a `List` class containing an array of string objects, write the method `int search (String oneToFind)` using binary search. This method uses the string parameter **oneToFind** and returns the index in the array if it is found, otherwise it returns -1 .
The binary search algorithm can be implemented as follows:

```
Given:  list, numItems, oneToFind

SET variables upperLimit and lowerLimit to their starting values
SET variable found to false
SET variable location to -1
WHILE (upperLimit hasn’t crossed over lowerLimit AND we haven’t found data)
        SET variable midpoint to middle between upperLimit and lowerLimit
        IF (list data at this midpoint is the data we are looking for)
                SET variable found to true
                SET variable location to midpoint
        ELSE
                IF (list data at this midpoint is greater than the data we are looking for)
                        // we know our data is between lowerLimit and midpoint
                        ADJUST upperLimit appropriately
                ELSE
                        // we know our data is between upperLimit and midpoint
                        ADJUST lowerLimit appropriately
                ENDIF
        ENDIF
ENDWHILE
RETURN location
```

```
public class List {
      private String [ ] list;
      private int numItems, maxSize;
      public  List() { list = null;  numItems = 0; maxSize = 0; }
      public void createList( ) {
                  // assume this code as been written properly…
      }
      public int search (String oneToFind){
                  // this is the code you are writing…
      }
      public String toString( ) {
                String display = new String (“The list is: “);
                for (int i = 0; i < nNumItems; i++)
                        display += “\n” + list[i];
                return display;
     }
}
```

```
public int search (String oneToFind) {
   int upperLimit = numItems -1;
   int lowerLimit = 0;
   int found = false;
   int location = -1;
   int mid;

   while (upperLimit < lowerLimit &&!found) {
      mid = (upperLimit + lowerLimit) /2;
      if (list[mid].compareTo(oneToFind) == 0) {
           // found
           found = true;
           location = mid;
           break;
      } else if (list[mid].compareTo(oneToFind) > 0)
           upperLimit = mid-1;
        else
           lowerLimit = mid+1;
   }
   return location;
}
```