Question 1 (10 marks)

Write a Java class `Student` with three fields: `name`, `mark` and `maxscore` representing a student who has scored `mark` out of `maxscore`. The class has two methods: `addBonus` adds a given bonus score to the student's mark, `toString` returns a String summarising the current state of the Student. You should select suitable types and parameters for your class its constructor and methods. Javadoc comments are **not** required for this question.

Answer Question 1 here

```
public class Student
{

private String name;
private int mark;
private int maxscore;

public Student(String name, int mark, int maxscore)
{
   if (mark > maxscore} { //bonus if some error handling }
   this.name = name;
   this.mark = mark;
   this.maxscore = maxscore;
}

public int addBonus(int bonus)
{
     mark = mark + bonus;
     if (mark > maxscore) { mark = maxscore; } //bonus for this
}

public String toString()
{
   return ("Stu: " + name + " has scored " + mark +
   " out of " + maxscore);
}
}
```

Question 2 (10 marks)

Using your Student class from Question 1, write an efficient method
`public Student topStudent(ArrayList<Student> cits1001)`
that returns the `Student` object with the highest percentage score from a given `ArrayList` of `Students` called `cits1001`. Use a helper method `percentMark` that returns the student's mark as a percentage rounded to the nearest integer. Javadoc comments are **not** required for this question.

Answer Question 2 here

```
private int percentMark()
{
     double pct = (double) mark * 100 / maxscore;
     //minor error for mark*100 / maxscore rounds down
     int mark = Math.round(pct);
     return ( mark );
}


public Student topStudent(ArrayList<Student> cits1001) {
       int max = cits1001.get(0).getPercentMark();
       Student best = cits1001.get(0);
        for (Student si : cits1001) {
            if (si.percentMark() > max) {
                max = si.percentMark();
                best = si;
            }
        }
        return best;
    }
```

Question 3 (10 marks)

Move tickets at the Windsor Cinema in Nedlands are priced as follows:

> Adult $16.50
> Subscriber Concession $12.00
> Senior / Pensioner / Children $11.00
> Tuesday is our cheap day at the Windsor Cinema:
> $8.50 before 6pm and $10.50 after 6pm on the cheap days

(a) Write a signature for a method `ticketPrice` to return the cost of a movie ticket (in cents) given the day of the week, session time, and customer status (Adult, Subscriber Concession, Senior, Pensioner, Child). Then design up to 5 test cases for this method, writing a JUnit assertEquals statement for each test.
(b) Write code to implement the `ticketPrice` method. Include a Javadoc header for the method that describes its parameters and return value. **You may assume that the given input parameters are all valid.** Javadoc comments **are** required for this question.

Answer Question 3(a) here

```
public int ticketPrice(String day, int session, String status)


Test cases such as
assertEquals(850, ticketPrice(2,1400,"Adult"));
assertEquals(1050, ticketPrice(2,1400,"Child"));
assertEquals(1100, ticketPrice(5,1400,"Senior"));
assertEquals(1200, ticketPrice(6,1400,"Subscriber Concession"));
assertEquals(1650, ticketPrice(1,1400,"Adult"));

NOT needed, illegal input checks
eg assertEquals(-1, ticketPrice("Anyday",0,"Martian"));
legal input check not needed
if ( (day<0) || (day > 6) || (hour > 2359)  ) { return -1; }
```

Answer Question 3(b) here

```
/**
* Calculate ticket price for Windsor cinema, assuming all inputs are legal
* @param day int day of week from 0 "Sunday" to 6 "Saturday"
* @param session int in 24 hour clock for the session time
* @param status String must be Adult, Subscriber Concession,
* Senior, Pensioner, Child to determine price
* @return int ticket price in cents
*/
public int ticketPrice(int day, int session, String status) {
a
if (day==2) {
   if (session < 1800) {
      return 850;
  }  else {
      return 1050;
   }
} else {
   if ( status.equals("Senior") ||
         status.equals("Pensioner") ||
         status.equals("Child") ) {
            return 1100;
    else if (status.equals("Subscriber Concession")) {
              return 1200;
       } else {
              return 1650;
       }
}
}
```

Question 4 (10 marks)

The tech-support project studied in labs and lectures mapped user query words to technical responses. Write a method, including a Javadoc header, with the signature

```
public String generateResponse(
                HashSet<String> words,
                HashMap<String, String> responseMap)
```

The method should return a selected response from its (already initialised) parameter `responseMap` linked to one of the words from the `HashSet` parameter. If no matching response is found then return `"Sorry I don't know"`.

Answer Question 4 here

```
/**
     * Generate a response from a given set of input words.
     *
     * @param words  A set of words entered by the user
     * @return       A string that should be displayed as the response
     */
     public String generateResponse(HashSet<String> words)
    {
        for (String word : words) {
            String response = responseMap.get(word);
            if(response != null) {
                return response;
            }
        }
        return "Sorry I don't know"; //otherwise
    }
```

Question 5 (10 marks)

Compare the methods `sort1` and `sort2` for sorting an array of integers. Comment on the strengths and weakness of each implementation using the criteria of readability, efficiency, cohesion and correctness.

```
public static void sort1(int[] a)
{
        for (int pass=1; pass<a.length; pass++) {
            int x=a[pass];
            int y=pass-1;
            while (y>=0 && a[pos]>x) {
                a[y+1]=a[y];
                y--;
            }
            a[y+1]=x;
        }
    }

public static void sort2(int[] a)
{
        for (int pass=1; pass < a.length; pass++) {
            for (int j = 0; j < a.length-pass; j++) {
                if (a[j] > a[j+1]) {
                     // exchange out of order elements
                     swap(a, j, j+1)
                 }
            }
        }
    }

private void swap(int[] a, int pos1, int pos2)
{
     int swap = a[pos2];
     a[pos2] = a[pos1];
     a[pos1] = swap;
}
```

Answer Question 5 on the next page

Answer Question 5 here

```
readability: sort1 does not use meaningful var name, sort2 does
sort2 includes a useful comment explaining why the swap

efficiency: sort2 is selection sort which is less efficient than
sort1 is insertion sort
because sort1 only moves part of the array

cohesion: sort2 uses a helper method which makes the main method
more readable and the method more cohesive

correctness: both are correct for non-null arrays.
both will crash if a is null (when a.length is called)
```

Question 6 (50 marks, 20 questions)

Question 6 must be answered on the machine readable multiple-choice answer sheet provided.

1. Which one of the following lines of Java code may cause a compilation error?

   *[a.] `String a = Hello;`

   b. `short a = 14;`

   c. `double a = 14;`

   d. `boolean a = true;`

   e. `int[] a = {1,2,3};`

2. Consider the Java variable declarations:

   ```
   int x = 7 / 5;
   int y = (-7) / 5;
   int z = 7 % 5;
   ```

   What are the values of `x`, `y` and `z` (respectively) after these declarations.

   a. 1, -1, 1

   b. 1.4, -1.4, 2

   *[c.] 1, -1, 2

   d. 1, -2, 2

   e. 1.4, -1.4, 2

3. You are writing a program for the Olympics Committee. The application has an `Athlete` class with fields to keep track of the name, gender, age, and sport of each athlete. What are the most appropriate types (respectively) to use for these variables?

   a. `String, String, double, String`

   b. `String, boolean, double, double`

   c. `String, String, int, double`

   *[d.] `String, char, int, String`

   e. `String, char, Integer, char[]`

4. What sort of variables are used to store the state of an individual object?

   a. Local variables.

   ***[b.]** Field variables.

   c. Class variables.

   d. Argument variables.

   e. Method variables.

5. The `Picture` class of the `shapes` project studied in lectures and labs includes the declaration `private Circle sun;`

   In the statement `sun.changeColor("yellow");` what does `changeColor` refer to?

   a. It is a field belonging to the class `Circle`.

   ***[b.]** It is a method belonging to the class `Circle`.

   c. It is a method belonging to the class `Picture`.

   d. It is a method belonging to the object `sun`.

   e. It is a field belonging to the object `sun`.

6. Which of the following statements is **not** true about information hiding in object oriented programming?

   a. It prevents "information overload"

   b. It reduces complexity by restricting access on a "need to know" basis.

   c. It is achieved, in part, by making instance variables private

   ***[d.]** It makes it harder to call the methods of a class because the implementation details are hidden.

   e. It makes it easier for teams of programmers to produce reliable software.

7. Consider the following block of code, where variables `a`, `b` and `c` each store integer values:

```
if (a < b) {
    if (a < c) {
        System.out.println(a);
    } else {
        System.out.println(c);
    }
} else if (b < c) {
    System.out.println(b);
} else {
    System.out.println(c);
}
```

   Which one of the following values for the variables `a`, `b` and `c` (respectively) will cause the value in variable `b` to be printed?

   a. 1, 2, 3

   b. 3, 2, 1

   c. 1, 3, 2

   ***[d.]** 2, 1, 3

   e. 2, 2, 1

8. What is the effect of the following Java statement

```
Die[] dice = new Die[10];
```

   a. A `Die` object representing a 10-sided die is formed

   b. An array containing 10 newly constructed `Die` objects is formed

   c. A label for an array is created, and initialized to the default value `null`

   ***[d.]** An array containing 10 `null` references is formed

   e. Ten labels, each referring to an array of objects of type `Die` are created.

9. The class `UniBook` has the following fields:

```
private String author;
private String title;
private int yearPublished;
private double cost;
```

Suppose that you are asked to write a method `computeTotalCost` that computes the total cost of books owned by each student. Which of the following method signatures would be most appropriate for the method?

a. `public void computeTotalCost()`

b. `private double computeTotalCost()`

c. `public double computeTotalCost(double[] booklist)`

d. `public double computeTotalCost(UniBook book)`

*[e.] `public double computeTotalCost(ArrayList<UniBook> booklist)`

10. What will the method call `sillyMethod(875)` return, where the method is defined as follows:

```
public int sillyMethod(int n) {
    int m = 0;
    while (n > 10) {
        m = n % 10;
        n = n / 10;
    }
    return m;
}
```

a. 875

b. 8

*[c.] 7

d. 5

e. 0

11. What output will the method call `show(5)` produce on the terminal window if the method `show` is defined as follows:

```
public void show(int n) {
  int num = 0;
  for (int i=1; i<n; i++)  {
    for (int j=0; j<i; j++) {
      System.out.print(num++);
      System.out.print(" ");
    }
    System.out.println(" ");
  }
}
```

a.
```
1
2 3
4 5 6
7 8 9 10
```

b,
```
0
0 0
0 0 0
0 0 0 0
```

c.
```
0
0 1
0 1 2
0 1 2 3
```

***[d.]**
```
0
1 2
3 4 5
6 7 8 9
```

e.
```
0
 1 2
  3 4 5
    6 7 8 9
```

12. Suppose that an array `int[] f` contains frequency data for some application. The *cumulative frequency* array is obtained by replacing each element in the array by the sum of itself and all of the previous elements in the array. In other words, the cumulative frequency array is `{f[0], f[0]+f[1], f[0]+f[1]+f[2], ...}`

    Which of the following three methods will correctly transform the frequency array `f` into a cumulative frequency array?

```
// Method 1
public void cumulate(int[] f)
{
  for (int i=1; i<f.length; i++) {
    f[i] = f[i] + f[i-1];
  }
}
// Method 2
public void cumulate(int[] f)
{
  for (int i=f.length-1; i>=0; i--) {
    for (int j=0; j<i; j++) {
      f[i] = f[i] + f[j];
    }
}
// Method 3
public void cumulate(int[] f)
{
  for (int i=0; i<f.length; i++) {
    for (int j=0; j<i; j++) {
      f[i] = f[i] + f[j];
    }
  }
}
```

    a. All of them.

    b. None of them.

    c. 2 and 3 only.

    d. 1 and 3 only.

    ***[e.]** 1 and 2 only.

13. What values would the variables `answer` and `words` hold after executing the following Java statements.

```
HashMap<String,String> translate = new HashMap<String,String>();
translate.put("hello","guten tag");
String answer = translate.get("goodbye");
int words = translate.size();
```

a. `""` and 1

b. `"guten tag"` and 1

***[c.]** `null` and 1

d. `null` and 2

e. No values because this code would cause a run-time exception to be thrown

14. Suppose that we execute the following code snippet

```
int[] a = new int[5];
int[] b = new int[5];

for (int i=0;i<5;i++) {
  a[i] = 2*i;
  b[i] = a[i];
}
```

What is the value of the expression `b == a`, and why does it have that value?

***[a.]** `false` because `a` and `b` are different arrays, even though they contain the same values.

b. `true` because `a` and `b` are arrays of the same length with the same values.

c. `true` because they are both arrays of the same type.

d. `false` because the values stored in the arrays are different.

e. `false` because `==` *always* returns `false` whenever reference types are compared.

15. Consider an incorrect implementation of a method `maxPos` in a utility class `AM`. The method is intended to return the index of the maximum element in its argument array:

```
public static int maxPos(int[] a)
   {
       if ((a == null)) {
           return -1;
       }
       int max = a[0];
       int maxpos = -1;
       for (int i = 0; i < a.length; i++) {
           if (a[i] > max) {
               max = a[i];
               maxpos = i;
           }
       }
       return maxpos;
   }
```

Which of the following JUnit4 tests will **fail** for this method?

1. `assertEquals(3,AM.maxPos(new int[] {5,7,1,10,-2,5}));`
2. `assertEquals(0,AM.maxPos(new int[] {5}));`
3. `assertEquals(4,AM.maxPos(new int[] {5,10,-2,5,11}));`
4. `assertEquals(-1,AM.maxPos(new int[] {}));`
5. `assertEquals(-1,AM.maxPos(null));`

a. 1, 4 and 5 only

b. 1 and 2 only

***[c.]** 2 and 4 only

d. 2, 3 and 4 only

e. all of them

16. Suppose that a class `ClassA` has a method with the signature `public void method()` and that variable `a` is declared and created in another class, `ClassB`, using: `ClassA a = new ClassA();`

    Consider the following three Java statements that occur in a method belonging to `ClassB`. Which one(s) are valid Java statement(s)?

    1. `ClassA.method();`
    2. `a.method();`
    3. `method();`

    a. 1 only

    b. Only 1 and 2

    ***[c.]** 2 only

    d. Only 1 and 3

    e. All of them

17. If a method uses checked exceptions to deal with unexpected (but recoverable) run-time events, how is that fact communicated to potential users of that method?

    a. In its return type

    b. In its name

    c. In its arguments

    ***[d.]** In its signature

    e. In its body

18. Which of the following collections would be most appropriate for storing a directory of names and phone numbers ?

    a. `ArrayList<String>`

    b. `HashSet<String>`

    ***[c.]** `HashMap<String, String>`

    d. `String[]`

    e. `HashMap<String, Integer>`

19. What does `mystery(a, b)` calculate, assuming `a` $\geq$ `0` and `b` $>$ `0`?

```
public int mystery(int a, int b)
{
    if (a < b) {
        return a;
     } else {
          return mystery(a - b, b);
     }
 }
```

a. It calculates `a + b`.

b. It calculates `a - b`.

c. It calculates `a * b`.

d. It calculates `a / b`.

***[e.]** It calculates `a % b`.

20. What happens when the statement
`int[] ra = reverse(new int[] {3,6,8,9});` is executed where

```
public int[] reverse(int[] a) {
        int[] rev = new int[a.length];
        for (int i=0; i < a.length; i++) {
            rev[i] = a[a.length  - i];
        }
        return rev;
    }
```

a. The array `ra` contains `{9,8,6,3}`

***[b.]** A `java.lang.ArrayIndexOutOfBoundsException` is thrown

c. A `java.lang.IllegalArgumentException` is thrown

d. The array `ra` contains `{8,6,3}`

e. None of the above

Blank Page for Rough Work