TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Student ID: .....

EXAMINATIONS — 2011

MID-YEAR

COMP 102
INTRODUCTION TO COMPUTER PROGRAM DESIGN

Time Allowed: 3 Hours ******** WITH SOLUTIONS **********

Instructions: Attempt ALL Questions.

Answer in the appropriate boxes if possible — if you write your answer elsewhere, make it clear where your answer can be found.

The exam will be marked out of 180 marks.

Non-programmable calculators without a full alphabetic key pad are permitted.

Non-electronic foreign language dictionaries are permitted.

Java Documentation will be provided with the exam script

There are spare pages for your working and your answers in this exam.

Questions

	Marks
1. Understanding Java	[57]
2. Files	[26]
3. Arrays of Objects	[26]
4. Interface classes	[13]
5. Event driven input	[15]
6. 2D Arrays	[23]
7. Debugging loops	[20]

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 1. Understanding Java [57 marks]

(a) [4 marks] What will the following fragment of Java print out?

```
int i = 1;
int j = 5;
while (j>0){
    UI.printf("%d %d %d\n", i, j, i*j);
    i++;
    j--;
}
```

```
1 5 5
2 4 8
3 3 9
4 2 8
5 1 5
```

(b) [5 marks] Almost every line of the following broken method has a syntax error that the compiler would complain about. Circle, and briefly explain each error

```
public int broken(number){                    // no type for the parameter
                 ^^^^^^^^
    if (number = 10){                         // need ==, not = for testing equality.
               ^^^
        UI.println("size 10);                 // missing double quote after 10
                          ^^^
        return;
    }          ^^^^^                          // does not return a value

    for (int i = 0, i < number, i++){
                  ^^^          ^^^            // should have ; not ,
        ans = ans + i;
         ^^^                                  // ans is not declared
    }
      ^^^^^^^^^^                              // no return statement
}
```

(Question 1 continued on next page)

(Question 1 continued)

(c) [6 marks] The printTimes method below has one parameter and prints out the times a store is open depending on the value of the parameter (note the **if**'s and **else**'s carefully):

```
public void printTimes(String today){
    if ( today.equals("saturday") || today.equals("sunday") ) {
        UI.print("Yes, we are open on ");
    }
    else if ( today.equals("thursday") ) {
        UI.print("Normal hours on ");
    }
    else if ( today.equals("monday") ) {
        UI.print("Open till late on ");
    }

    UI.println(today);

    if ( today.startsWith("t") ) {
        UI.println(today + " special sale");
    }
    else {
        UI.println(today + " half day");
    }
}
```

What would the following calls to printTimes print out?

printTimes("saturday"); ⟹
yes, we are open on saturday
saturday half day

printTimes("thursday"); ⟹
normal hours on thursday
thursday special sale

printTimes("MONDAY"); ⟹
MONDAY half day

(Question 1 continued on next page)

(Question 1 continued)

(d) [5 marks] Complete the thirdAngle method below which should compute the size of the third angle of a triangle. It should have two parameters (doubles) which are the sizes of the first two angles of a triangle.
Hint: the value of the third angle is 180 minus the sum of the first two angles.

```
public double thirdAngle (double angle1, double angle2){

    return 180 − (angle1+angle2);


}
```

(e) [5 marks] Consider the following printSeries method.

```
public void printSeries( int max){
    int prev = 1;
    UI.print(prev + "  ");
    int curr = 2;
    while(curr <= max){
        UI.print(curr + "  ");
        int next = 2*curr + prev;
        prev = curr;
        curr = next;
    }
    UI.println("Done");
}
```

What will be printed in the text pane if printSeries is called with the argument 30?

```
1  2  5  12  29  Done
```

(Question 1 continued on next page)

(f) [7 marks] Consider the following printStuff method which copies some values of an array of String into another array, then prints out the other array.

```
public void printStuff (){
    String [] animals = new String[]{"cat","tiger","dog","turtle","sheep","eel","snake"};
    String [] pets = new String [animals.length];

    int count=0;
    for( int i=0; i<animals.length; i++){
        if (animals[i ]. length()>3){
            pets[count] = animals[i ];
            count++;
        }
    }

    for( int i=0;i<pets.length;i++){
        if (pets[ i ]!=null){
            UI. println (pets[ i ]);
        }
        else {
            UI. println ("--");
        }
    }
}
```

What will be printed in the text pane if printStuff is called?
Hint: show your working, especially the contents of the pets array.

```
tiger
turtle
sheep
snake
-
-
-
```

(Question 1 continued on next page)

(Question 1 continued)

(g) [7 marks] Suppose the file `offices.txt` contains the following text:

```
Anderson 323 Cotton
Bayson   518 Cotton
Carson   841 Kirk
```

What will the following printOffices method print out?

```
public void printOffices (){
    try{
        Scanner scan = new Scanner (new File("offices.txt"));
        int count = 0;
        while ( scan.hasNext() ){
            if (scan.hasNextInt()){
                count = scan.nextInt ();
            }
            else {
                String name = scan.next();
                UI. println ("name: " + name);
            }
        }
        UI. println ("Read " + count + " offices ");
        scan.close();
    }
    catch(IOException e){UI.println("File operation failed");}
}
```

```
name Anderson
name Cotton
name Bayson
name Cotton
name Carson
name Kirk
Read 841 offices
```

(Question 1 continued on next page)

(h) [8 marks] The Sushi2Go class on the facing page defines Sushi2Go objects, which have three fields to store their fillType, traySize, and sushiCount. The class defines a constructor and three methods.

What will the following testSushi method print out?

```
public void testSushi(){
    Sushi2Go order1 = new Sushi2Go("fish", 6);
    UI. println (order1.toString ());

    order1.addPieces(5);
    UI. println (order1.toString ());
    order1.addPieces(4);
    UI. println (order1.toString ());

    Sushi2Go order2 = new Sushi2Go("eel", 4);
    order2.addPieces(3);
    UI. println (order2.toString ());

    order1.swapTray(order2);
    UI. println (order1.toString ());

    Sushi2Go order3 = new Sushi2Go("carrot", 10);
    order3.addPieces(3);
    order3.swapTray(order1);
    UI. println (order3.toString ());
}
```

```
0 of fish on tray 6

5 of fish on tray 6

6 of fish on tray 6

3 of eel on tray 4

Swap failed

6 of fish on tray 6

3 of carrot on tray 6
```

(Question 1 continued on next page)

(Question 1 continued)

```
public class Sushi2Go{
    private String fillType ;
    private int traySize;      // maximum number of pieces
    private int pieces;        // current number of pieces in the tray

    public Sushi2Go(String filling , int size){
        this. fillType = filling ;
        this.traySize = size;
        this.pieces = 0;
    }

    public String toString(){
        return (this.pieces + " of " + this.fillType + " on tray " + this.traySize);
    }

    public void addPieces (int n){
        this.pieces = Math.min(this.pieces + n, this.traySize);
    }

    public void swapTray(Sushi2Go other){
        if ( this.pieces < other.traySize && other.pieces < this.traySize){
            int temp = this.traySize;
            this.traySize = other.traySize;
            other.traySize = temp;
        }
        else {
            UI. println ("Swap failed");
        }
    }

}
```

(Question 1 continued on next page)

(Question 1 continued)

(i) [10 marks] Complete the SpaceShip class on the facing page that stores information about space ships.

A SpaceShip object should contain two fields:

- shipName, which contains the name of the ship.
- planetsVisited, which contains the number of planets the ship has been on. The initial value of the field should be 1.

SpaceShip should have a constructor that takes one String parameter and stores it in the shipName field.

SpaceShip should have three methods:

- getPlanets(), which returns the number of planets the SpaceShip has visited.
- explorePlanet(), which is called when the SpaceShip visits a new planet, and should update the planetsVisited field to record that the ship has visited one more planet.
- toString(), which returns a String containing the name of the space ship and the number of planets visited in the form `"VogonExplorer has visited 35 planets"`

(Question 1 continued on next page)

(Question 1 continued)

```
public class SpaceShip{
  // fields
    private String shipName;
    private int planetsVisited = 1;


  // constructor
    public SpaceShip(String sName){
        this.shipName = sName;
    }



 // methods


    public int getPlanets(){
       return this.planetsVisited;
    }

    public void explorePlanet(){
       this.planetsVisited++;
    }

    public String toString(){
        return (this.shipName + " has visited " + this.planetsVisited + " planets");
    }









}
```

Question 2. Files [26 marks]

Suppose the file `MemorySticks.txt` contains data about memory sticks. Each line of the file contains information about memory stick - the manufacturer, capacity (GB), speed(MB/s), the model, and the price. For example, the following might be the first eight lines of the file.

```
SanDisk       8    3   Pro Duo             38.82
SanDisk       8   10   Ultra               79.95
SanDisk      16    4   Pro Duo Gaming      78.89
Lexar        32    9   Jump Drive Retrax   59.46
Sony         16   20   Pro-HG Duo         103.85
Sony          4    4   Pro Duo Mark2       33.89
Transcend     8    5   Pro Duo             70.53
```

The manufacturer name is always a single word; the model name consists of one or more words.

(a) [7 marks] Complete the following listManufacturer method, whose parameter is the name of a manufacturer (*e.g.* `Lexar`). listManufacturer should print out the details of each memory stick in the file of the given manufacturer.

For example, on the data above, listManufacturer("Sony"); should print

```
Memory Sticks from Sony
    16   20   Pro-HG Duo     103.85
     4    4   Pro Duo Mark2   33.89
```

```
public void listManufacturer(String mfr){
    String filename = "MemorySticks.txt";
    UI.println("Memory Sticks from "+ mfr);
    try{
        Scanner sc = new Scanner(new File(filename));
        while (sc.hasNext()){
            String m = sc.next();
            String restOfLine = sc.nextLine();
            if (m.equals(mfr)){
                UI.println(restOfLine);
            }
        }
        sc.close();









    }catch(IOException e){UI.println("File operation failed: " + e);}
}
```

(Question 2 continued on next page)

(Question 2 continued)

(b) [9 marks] Complete the following maxSpeed method which should print out the manufacturer that makes the fastest memory stick in the file.

For example, if the file had only the eight lines in the example above, maxSpeed() should print out "Fastest is Sony" because Sony makes the fastest memory stick (the 20Mb/s one).

```
public void maxSpeed(){
    String filename = "MemorySticks.txt";
    String
    try{
        Scanner sc = new Scanner(new File(filename));
         int maxSpeed = 0;
         String maxManufacturer = "";
         while (sc.hasNext()){
              String mfr = sc.next();
              int cap = sc.nextInt();
              int speed = sc.nextInt();
              String rest = sc.nextLine();
              if (speed > maxSpeed){
                        maxSpeed = speed;
                        maxManufacturer = mfr;
              }
         }
         UI.println("Fastest is " + maxManufacturer);



        sc.close();
    }
    catch(IOException e){UI.println("File operation failed: " + e);}
}
```

(Question 2 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 2 continued)

(c) [10 marks] Complete the following cheapModels method which is passed a price, and then prints out the manufacturer, model name, and capacity for all the memory sticks less than that price.

For example, if the file had the eight lines in the example above, cheapModels(60.0) should print:

```
SanDisk Pro Duo 8GB
Lexar Jump Drive Retrax 32GB
Sony Pro Duo Mark2 4GB
```

since these are all the memory sticks under $60.00.

```
public void cheapModels(double maxPrice){
    String filename = "MemorySticks.txt";
    try{
        Scanner sc = new Scanner(new File(filename));
        while (sc.hasNext()){
            String mfr = sc.next();
            int cap = sc.nextInt();
            int speed = sc.nextInt();
            String model = sc.next();
            while (!sc.hasNextDouble()){
                model = model +"  " + sc.next();
            }
            double price = sc.nextDouble();
            if (price < maxPrice){
                UI.println(mfr+"  "+model+"  "+cap+"GB");
            }
        }
        sc.close();
    }
    catch(IOException e){UI.println("File reading failed: " + e);}
}
```

Question 3. Arrays of Objects [26 marks]

This question concerns a CricketAnalyser program to analyse the performance of cricket batsmen. The CricketAnalyser class (on the facing page) stores the information about the batsmen in a field containing an array of Batsman objects. It also has a count field that contains the number of Batsman objects, and the Batsman objects are stored in cells 0 through count-1 of the array.

The Batsman class, for representing information about individual batsmen, is shown below.

```
public class Batsman{

    private String name;
    private String country;
    private int gamesPlayed;
    private double battingAvg;

    public Batsman(String nm, String ctry, int games, double avg){
        this.name = nm;
        this.country = ctry;
        this.gamesPlayed = games;
        this.battingAvg = avg;
    }
    public void printDetails (){
        UI. printf ("%s of %s av: %.2f over %d\n",
                this.name, this.country, this.battingAvg, this.gamesPlayed );
    }
    public boolean hasName(String nm){
        return (this.name.equals(nm));
    }
    public String getCountry(){
        return this.country;
    }
    public int getGamesPlayed(){
        return this.gamesPlayed;
    }
    public double getBattingAvg(){
        return this.battingAvg;
    }
}
```

.

(Question 3 continued)

The following are some of the fields and two of the methods of the CricketAnalyser class:

```
public class CricketAnalyser{

    private Batsman [ ] cricketers  = new Batsman [8000];
    private  int  count = 0;

    public void  listCricketers (){
        for  ( int  i=0; i<this.count; i++){
            this. cricketers [ i ]. printDetails  ();
        }
    }
    public Batsman findCricketer(String name){
        for  ( int  i=0; i<this.count; i++){
            if  (this. cricketers [ i ]. hasName(name) ){
                return this. cricketers [ i ];
            }
        }
        return null;
    }
```

(a) [6 marks] Complete the following addCricketer method of the CricketAnalyser class. Its parameter is a Batsman and it should add the given Batsman to the cricketers array. If the array is full, the method should print out a message.

```
public void addCricketer (Batsman batsman){
    if  (this.count < this.cricketers .length)  {
        this. cricketers [this.count++] = batsman;
    }
    else {
        UI. println ("No room to add");
    }

}
```

(Question 3 continued on next page)

(b) [6 marks] Complete the following findHighest method which should find the batsman with the highest batting average and print out their details (using the printDetails method). If there is a tie for highest, it may print any of the highest batsmen.

```
public void findHighest(){
    int highest = 0;
    for (int i=0; i<this.count; i++){
        if (this.cricketers[i].getBattingAvg() > this.cricketers[highest].getBattingAvg()){
            highest = i;
        }
    }
    this.cricketers[highest].printDetails();









}
```

(c) [6 marks] Complete the following longServers method which has one parameter — the name of a country — and should print the details of all the cricketers from the specified country who have played more than 100 games.

```
public void longServers(String country){
    for (int i=0; i<this.count; i++){
        if (country.equals(this.cricketers[i].getCountry())&&
            this.cricketers[i].getGamesPlayed() > 100) {
            this.cricketers[i].printDetails());
        }
    }









}
```

(Question 3 continued on next page)

(Question 3 continued)

(d) [8 marks] Complete the following boycott method in the CricketAnalyser class. Its parameter is *String* specifying a country, and it should remove *all* cricketers from this country from the list of cricketers. It is important that there should be no nulls in the cricketers array in the range from 0 to count-1. The order the cricketers are stored in the array is not important and may be changed.

```
public void boycott(String country){
    for (int i=0; i<this.count; i++){
        if (country.equals(this.cricketers[i].getCountry())){
            count--;
            this.cricketers[i] = this.cricketers[count];
            i--;    // in order to check the cricketer we have just moved down
        }
    }
}
OR // saves rechecking because we always move down one that has been checked
    public void boycott(String country){
    for (int i=this.count-1; i>=0; i--){
        if (country.equals(this.cricketers[i].getCountry())){
            count--;
            this.cricketers[i] = this.cricketers[count];
        }
    }
}
```

Question 4. Interface classes [13 marks]

The CricketAnalyser program in question 3 involved an array storing a collection of Batsman objects. Each Batsman had several fields, including the name, country, and number of games they had played.
Note, this question is independent of your answers for question 3.

The CricketAnalyser program is limited because it only stored and analysed information about batsmen; a cricket program should also work with information about bowlers, wicket keepers and fielders. For this question, you need to modify the CricketAnalyser program so that it can store a collection of different kinds of Cricketers — Batsman, Bowler, WicketKeeper, and Fielder.

All four kinds of Cricketer have these things in common:

- fields to store a name, country, number of games
- a printDetails method to print out information about the Cricketer
- a role method that returns a String that is the Cricketer's primary role ("Batsman", "Bowler", or "WicketKeeper").
- a hasName method with one parameter that returns a boolean value saying whether the Cricketer has the specified name.
- getCountry and getGamesPlayed methods which return the Cricketer's country and number of games played, respectively.

The four different kinds of Cricketer each have additional fields and methods, specific to their role.

(a) [5 marks] Complete the following definition of the Cricketer interface class to represent the Cricketer type:

```
public interface Cricketer {
    public void printDetails ();

    public String role ();

    public boolean hasName(String name);

    public String getCountry();

    public int getGamesPlayed();

}
```

(Question 4 continued on next page)

(Question 4 continued)

(b) [4 marks] Modify the Batsman class below so that a Batsman is a type of Cricketer, ensuring that it correctly implements the Cricketer interface class.
Note: The code below is identical to the code on page 16 for Question 3.

```
public class Batsman { XX   implements Cricketer {

    private String name;
    private String country;
    private int gamesPlayed;
    private double battingAvg;

    public Batsman(String nm, String ctry, int games, double avg){
        this.name = nm;
        this.country = ctry;
        this.gamesPlayed = games;
        this.battingAvg = avg;
    }

    public void printDetails (){
        UI. printf ("%s of %s av: %.2f over %d\n",
                this.name, this.country, this.battingAvg, this.gamesPlayed );
    }

    <<<<<<<<<<<<<  public String role () { return "Batsman"; }
    public boolean hasName(String nm){
        return (this.name.equals(nm));
    }
    public String getCountry(){
        return this.country;
    }
    public int getGamesPlayed(){
        return this.gamesPlayed;
    }
    public double getBattingAvg(){
        return this.battingAvg;
    }

}
```

(Question 4 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 4 continued)

(c) [4 marks] Modify the part of the CricketAnalyser program shown below so that it can store a collection of any type of Cricketer. You may or may not need to modify the two method definitions. Note: The code below is identical to the code on page 17 for Question 3.

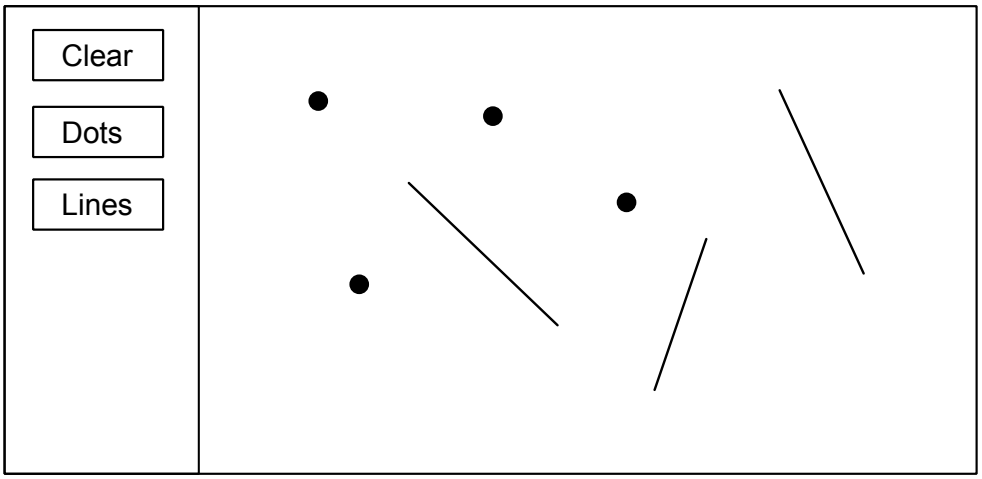
```
public class CricketAnalyser{
                Cricketer                       Cricketer
    private Batsman [] cricketers  = new Batsman [8000];
    private  int  count = 0;

    public void  listCricketers (){
        for  ( int  i=0; i<this.count; i++){
            UI. print ( i+": ");
            this . cricketers [ i ]. printDetails ();
        }
    }
            Cricketer
    public Batsman findCricketer(String name){
        for  ( int  i=0; i<this.count; i++){
            if  (this . cricketers [ i ]. hasName(name) ){
                return this . cricketers [ i ];
            }
        }
        return null ;
    }


       :


}
```

Question 5. Event Driven Input [15 marks]

Complete the DotAndLine program on the facing page so that it allows the user to draw a pattern of round dots and straight lines on the graphics pane. The program should have three buttons labeled "Clear", "Dots", and "Lines". The "Clear" button should clear the graphics pane. The other two buttons control the effect of the mouse. After the user has clicked the "Dots" button, the program should draw a small circle (5 units wide) at each point that the user releases the mouse. After the user has clicked the "Lines" button, the program should draw a straight line between the point that the user pressed the mouse and the point the user releases the mouse.

The following diagram shows the effect of

- clicking the Clear button,
- clicking the Dots button,
- releasing the mouse at four points to make four dots
- clicking the Lines button,
- dragging the mouse three times to make three lines.



(Question 5 continued)

```
public class DotAndLine implements UIButtonListener, UIMouseListener{

    private double lastX = -1;
    private double lastY = -1;
    private String shape = "Dots";

    public DotAndLine(){
        UI.setMouseListener(this);
        UI.addButton("Dots", this);
        UI.addButton("Lines", this);
        UI.addButton("Clear", this);


    }
    public void buttonPerformed(String button){
        if (button.equals("Dots")){
            this.shape = "Dots";
        }
        else if (button.equals("Lines")){
            this.shape = "Lines";
        }
        else if (button.equals("Clear")){
            UI.clearGraphics();
        }



    }
    public void mousePerformed(String action, double x, double y) {

        if (action.equals("pressed")){
            this.lastX = x;
            this.lastY = y;
        }
        else if (action.equals("released")){
            if (this.shape.equals("Dots")){
                UI. fillOval (x-2, y-2, 5, 5);
            }
            else{
                UI.drawLine(this.lastX, this.lastY, x, y);
            }
        }


    }
}
```

Question 6. 2D Arrays [23 marks]

A Word Find puzzle is a 2D array of letters that contains words hidden in the rows and columns. For example, the following Word Find contains the word "CUBA" hidden in row 2 and the word "MALL" in column 4.

D	W	K	I	O
Q	S	A	D	**M**
X	**C**	**U**	**B**	**A**
J	H	R	D	**L**
M	E	O	P	**L**

The WordFinder class implements a Word Find puzzle. It uses a field containing a 2D array of Strings (all of which will be a single letter) to store a Word Find puzzle. You are to complete several methods for the WordFinder class.

```
public class WordFinder{

    private String[ ][ ] puzzle;

      :
```

(a) [6 marks] Complete the following printPuzzle method which should print out a puzzle in the textPane with a space between each letter. For example, if the puzzle field contained the word find puzzle above, printPuzzle would print out

```
D W K I O
Q S A D M
X C U B A
J H R D L
M E O P L
```

```
private void printPuzzle(){
    for (int row = 0; row < this.puzzle.length; row++) {
        for (int col = 0; col < this.puzzle[row].length; col++) {
            UI.print(this.puzzle[row][col]+" ");
        }
        UI.println();
    }




}
```

(Question 6 continued on next page)

(Question 6 continued)

(b) [7 marks] Complete the following loadPuzzle method which should read a puzzle from a file into the puzzle field. The parameter of the method is the name of the file to load from. The first line of the file will contain two integers: the number of rows and columns of the puzzle. The remaining lines will contain all the letters in the puzzle, separated by spaces, with the letters in the first row first, followed by the second row, *etc*.

For example, a file with the following contents would represent the 5x5 puzzle above:

```
5 5
D W K I O Q S A D M X C U B A
J H R D L M E O P L
```

```
public void loadPuzzle(String fname) {
    try{
        Scanner file = new Scanner(new File(fname));
        this.puzzle = new String[file.nextInt()][file.nextInt()];
        for(int row = 0; row < this.puzzle.length; row++){
            for(int col=0; col < this.puzzle[row].length; col++){
                this.puzzle[row][col] = file.next();
            }
        }









        file.close();
    }catch(IOException e){UI.println("File reading failed" + e);}
}
```

(Question 6 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 6 continued)

(c) [10 marks] Complete the following findWordHorizontal method which should search for a word hidden in a row of a puzzle. The parameter of the method is an array containing the word, one String for each letter. The method must search each row for the word. If it finds the word, it will print out the row and column where the word starts.

For example, with the puzzle above, the following code

```
String[ ] wordToFind = new String[ ] {"C", "U", "B", "A"};
findWordHorizontal(wordToFind);
```

should print out `"Found at (2,1)"` since the word `CUBA` starts at row 2, column 1.

```
public void findWordHorizontal(String [ ] word){
    for( int row = 0; row < this. letters .length; row++) {
        for( int col = 0; col < this. letters [row].length−word.length+1; col++){
            boolean found = true;
            for ( int i=0; i<word.length; i++){
                if (!word[i ]. equals(this. letters [row][col+i ])){
                    found = false;
                    break;
                }
            }
            if (found){
                UI. printf ("Found at (%d, %d)\n", row, col);      );
                displayHorizontal(row, col, word.length);
            }
        }
    }
}
```

Question 7. Debugging Loops [20 marks]

The following moveSmallestToFront method is intended to find the minimum value in an array of numbers, and swap all copies of that value to the front of the array.

For example, given the array

6	4	0	7	-2	7	-1	-2	7	-2
0	1	2	3	4	5	6	7	8	9

moveSmallestToFront should find that -2 is the minimum value, and then swap all the -2's to the front:

-2	-2	-2	7	6	7	-1	4	7	0
0	1	2	3	4	5	6	7	8	9

This version of moveSmallestToFront has multiple errors.

```
1   public void moveSmallestToFront(double[] data){
2       double min = 0;              // error – need to start large or data[0]
3       for (int i=0; i<data.length; i++){
4           if (min < data[i]){      // error inverted comparison
5               min = data[i];
6           }
7       }
8       int indexInsert = 0;
9       for (int i=0; i<data.length-1; i++){
10          if (data[i] == min){
11              data[i] = data[indexInsert++];  // error, need to increment
12              data[indexInsert] = min;        // after swapping
13          }
14      }
15  }
```

(a) [8 marks] If this version of moveSmallestToFront (with the errors) was passed the first array above, what would the contents of the array be when the method was finished?

6	7	7	7	-2	7	-1	-2	7	-2
0	1	2	3	4	5	6	7	8	9

(Question 7 continued on next page)

(Question 7 continued)

(b) [6 marks] Circle and briefly describe three errors in the version of moveSmallestToFront above.
You may answer on the code above or list line numbers and describe the error in the box below

line 2: need to start with a large number or data[0]
line 4: the comparison is inverted
line 11: need to increment indexInsert after swapping

(c) [6 marks] Write a correct version of moveSmallestToFront.

```
public void moveSmallestToFront(double[] data){
    double min = data[0];
    for (int i=0; i<data.length; i++){
        if (data[i]< min){
            min = data[i];
        }
    }
    int indexInsert = 0;
    for (int i=1; i<data.length; i++){
        if (data[i] == min){
            data[i] = data[indexInsert];
            data[indexInsert++] = min;
        }
    }

}
```

****