

THE UNIVERSITY OF AUCKLAND

Second Semester, 2003
City/Tamaki Campus

COMPUTER SCIENCE

Principles of Programming

(Time allowed: TWO HOURS)

Surname:	
Forenames:	
Student ID number:	
Login name (UPI):	

INSTRUCTIONS:

- Attempt **ALL** questions - write your answers in the box provided
- Calculators are **NOT** permitted

Examiner to complete:

Question	Mark
1	(/10)
2	(/20)
3	(/4)
4	(/10)
5	(/12)

Question	Mark
6	(/12)
7	(/8)
8	(/4)
9	(/10)
10	(/10)

TOTAL:

(/100)

CONTINUED

SURNAME: FORENAMES:

Question 1 (10 marks)

- (a) What is printed when the following statement is executed?

```
System.out.println( 1 + 2.5 + "4" + 3/2.0 );
```

(1 mark)

- (b) What is printed when the following statement is executed?

```
System.out.println( 103/10*10 );
```

(1 mark)

- (c) What is printed when the following statement is executed?

```
System.out.println( 103%10 );
```

(1 mark)

- (d) What is printed when the following statement is executed?

```
System.out.println( (true && false) || (true ^ true) );
```

(1 mark)

SURNAME: FORENAMES:

- (e) What is printed when the following statement is executed?

```
System.out.println( (true || (true && false)) || false );
```

(1 mark)

- (f) What is printed when the following statement is executed?

```
System.out.println( 2 > 3 && !(3 - 2 < 1 || 1 == 2) );
```

(1 mark)

- (g) Convert the following formula into a single Java statement. You may assume that the variables *area* and *r* have already been declared as doubles.

$$\text{area} = 4\pi.r^2$$

(2 marks)

- (h) Convert the following formula into a single Java statement. You may assume that the variables *a*, *b* and *c* have already been declared as doubles.

$$c = \sqrt{a^2 + b^2}$$

(2 marks)

SURNAME: FORENAMES:

Question 2 (20 marks)

Read the following program and answer the questions below. Try to understand the purpose of the code.

```
import java.io.*;

public class Question02{
    public static void main(String[] args){
        System.out.print("Enter a number: ");
        int n = Integer.parseInt(readInput());
        boolean result = check(n);
        System.out.println("Result: " + result);
    }

    private static boolean check(int n){
        System.out.println("check: " + n);
        while(n!=0){
            int om = getOM(n);
            int a = getMS(n, om);
            int b = getLS(n);
            n = (n%om)/10;
            if(a!=b)
                return false;
        }
        return true;
    }

    private static int getOM(int n){
        int i=1;
        while(i<=n){
            i=i*10;
        }
        i = i/10;
        System.out.println("getOM return: " + i);
        return i;
    }

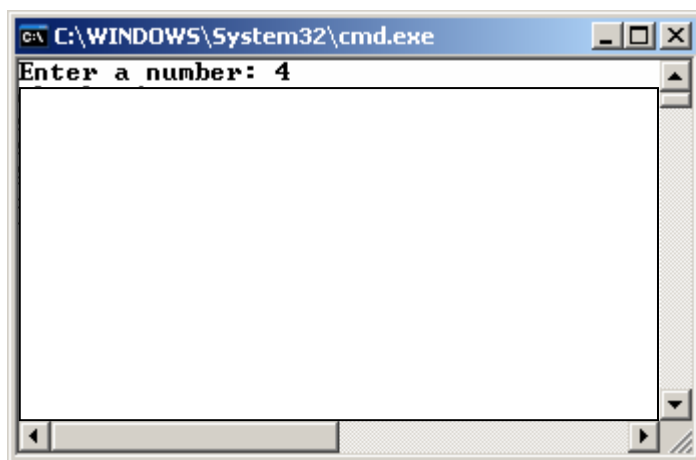
    private static int getMS(int n, int om){
        System.out.println("getMS return: " + n/om);
        return n/om;
    }

    private static int getLS(int n){
        System.out.println("getLS return: " + n%10);
        return n%10;
    }

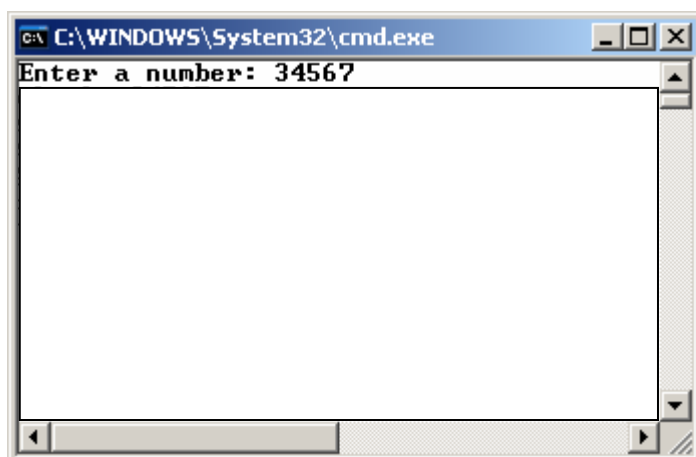
    private static String readInput(){
        //Assume code for readInput method is include here
    }
}
```

SURNAME: FORENAMES:

- (a) What is the output produced by the program when the user enters the number 4? Enter your answer in the box below.

*(5 marks)*

- (b) What is the output produced by the program when the user enters the number 34567? Enter your answer in the box below.

*(5 marks)*

- (c) Write a comment which explains the purpose of the method with the signature given below. Your comment should be brief (i.e. only one or two lines).

```
/*  
  
  
  
  
  
  
  
  
  
*/  
private static int getOM(int n){
```

*(2 marks)***CONTINUED**

SURNAME: FORENAMES:

- (d) Write a comment which explains the purpose of the method with the signature given below. Your comment should be brief (i.e. only one or two lines).

```
/*  
  
*/  
private static int getLS(int n){
```

(2 marks)

- (e) Give an example of an integer number that the user could enter, which would result in a run-time error being generated.

(2 marks)

- (f) Give an example of a 5 digit number which when used as input would cause the line "Result: true" to be printed.

(2 marks)

- (g) Complete the comment below which explains the purpose of this program. Your answer should be brief (i.e. only one or two lines).

```
/*  
Author: Andrew Luxton  
Date: 13/10/2003  
Purpose:  
  
  
  
  
  
  
  
  
  
*/  
public class Question02{
```

(2 marks)

SURNAME: FORENAMES:

Question 3 (4 marks)

Write the output that will be printed by the following code fragment.

```
int[] shuffled = { 9, 2, 1, 10, 7, 5, 3, 12, 11, 0, 4, 6, 8 };
int draw;
String[] card = { "Ace", "2", "3", "4", "5", "6", "7", "8", "9",
                  "10", "Jack", "Queen", "King" };
for ( draw = 0; draw < 5; draw = draw + 1 )
    System.out.print( card[shuffled[ draw ]] + "  " );
System.out.println();
```

(4 marks)

SURNAME: FORENAMES:

Question 4 (10 marks)

Write the method, `indexOfClosest()`. This method is passed two parameters, an array of `ints` and an `int` number. The method returns the INDEX of the number in the array which is closest in value to the number passed in as a parameter. If there is more than one number in the array which is equally close in value to the parameter number then the method returns the index of the FIRST number in the array which is closest to the parameter number (i.e. closest to the front of the array).

Notes:

- You can assume that the array passed as a parameter is not null, and that it contains at least one element.
- You may find the `Math.abs()` method useful in this question.

Example: An example showing the use of the `indexOfClosest` method is included below. The code makes use of a method called `printArray()`. The `printArray()` method is not shown below. When the following code is executed with the completed `indexOfClosest()` method:

```
int[] array1 = {20, -3, 6, 9, 6, -5};
System.out.print("array1: ");
printArray(array1);
System.out.println();

int indexOfClosest = indexOfClosest(array1, 7);
System.out.println("The index of the number closest to 7 is: "
                  +indexOfClosest);
System.out.println("In the array the closest number to 7 is: "
                  +array1[indexOfClosest]);
System.out.println();

indexOfClosest = indexOfClosest(array1, 40);
System.out.println("The index of the number closest to 40 is: "
                  +indexOfClosest);
System.out.println("In the array the closest number to 40 is: "
                  +array1[indexOfClosest]);
System.out.println();

indexOfClosest = indexOfClosest(array1, -3);
System.out.println("The index of the number closest to -3 is: "
                  +indexOfClosest);
System.out.println("In the array the closest number to -3 is: "
                  +array1[indexOfClosest]);
```

the output is:

```
array1: 20 -3 6 9 6 -5

The index of the number closest to 7 is: 2
In the array the closest number to 7 is: 6

The index of the number closest to 40 is: 0
In the array the closest number to 40 is: 20

The index of the number closest to -3 is: 1
In the array the closest number to -3 is: -3
```

SURNAME: FORENAMES:

```
private static int indexOfClosest(int[] nums, int number){
```

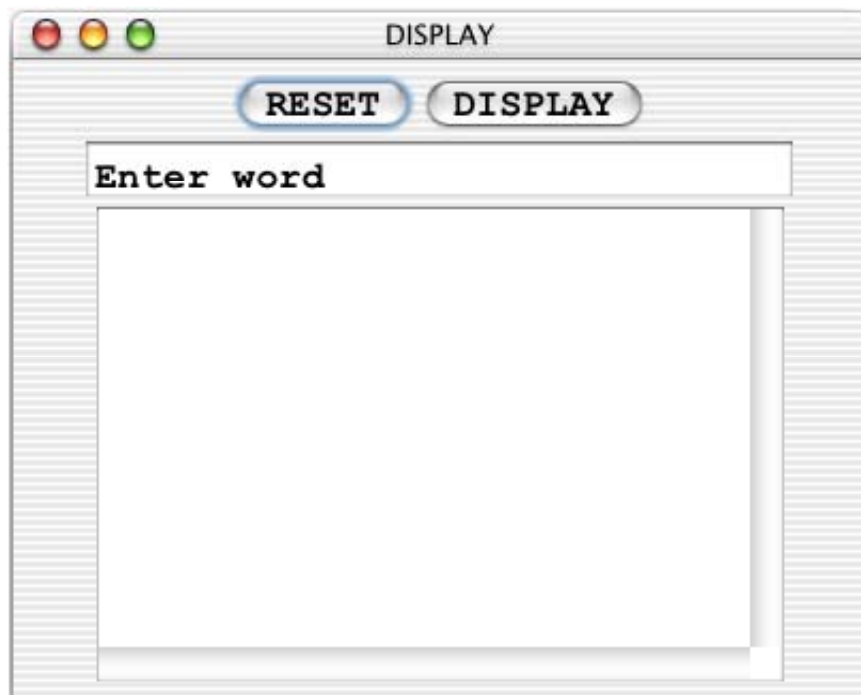
```
}
```

*(10 marks)***CONTINUED**

SURNAME: FORENAMES:

Question 5 (12 marks)

You are required to complete the `DisplayFrame` definition. The `DisplayFrame` contains two Buttons, one `TextField` and one `TextArea`. When the Frame first appears it looks like the following screenshot. The `TextField` displays the text "Enter word":

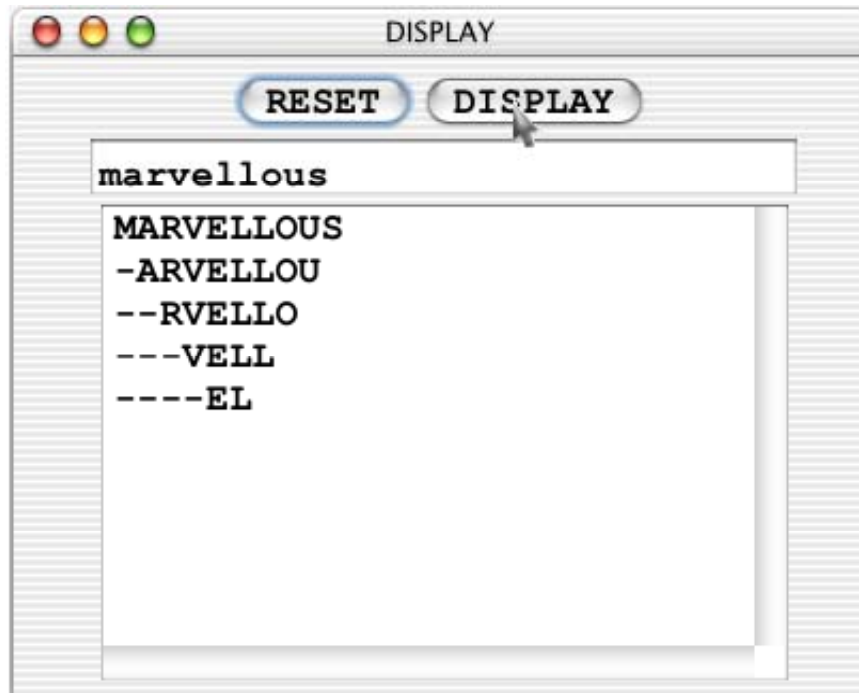


Whenever the user presses the `RESET` Button, the Frame should have the same appearance as when the Frame is first created i.e. the same as shown in the screenshot above.

Whenever the user enters a word in the `TextField` and presses the `DISPLAY` Button the word from the `TextField` is displayed in the `TextArea` in a series of lines. Each line of the display is shortened by removing the first and the last character until the word contains no more characters. The left indent is made up of an increasing number of '-' characters. The characters of the display are all in uppercase characters.

Look at the sample screenshots shown below. In the first screenshot the user has entered the word "marvellous" and pressed the `DISPLAY` Button, and in the second screenshot the user has entered the word "fantastic" and pressed the `DISPLAY` Button.

SURNAME: FORENAMES:



SURNAME: FORENAMES:

```
import java.awt.*;
```

```
public class DisplayFrame extends Frame
```

```
    //Buttons which are added to the Frame
    private Button displayB, resetB;
```

```
    //TextField which contains the word to be displayed
    private TextField wordT;
```

```
    //TextArea used to display the word
    private TextArea displayTA;
```

```
    public DisplayFrame(String title, int x, int y, int width, int height){
        setBounds(x,y,width,height);
        setTitle(title);
        addWindowListener(new WindowAdapter(){
            public void windowClosing(WindowEvent e){
                dispose();
                System.exit(0);
            }
        });

        setLayout(new FlowLayout());
        addAllComponents();
        reset();
        show();
    }
```

```
    private void addAllComponents(){
        resetB = new Button("RESET");
        add(resetB);
```

```
        displayB = new Button("DISPLAY");
        add(displayB);
```

```
        wordT = new TextField(30);
        add(wordT);
```

```
        displayTA = new TextArea(10,30);
        add(displayTA);
    }
```

SURNAME: FORENAMES:

```
public void actionPerformed(ActionEvent e){
```

```
}
```

```
private void reset(){
```

```
}
```

```
private void displayWord(String word){
```

```
}
```

```
}
```

*(12 marks)***CONTINUED**

SURNAME: FORENAMES:

Question 6 (12 marks)

Complete the `GiftVoucher` class definition. The `GiftVoucher` class stores information (the id number, the value of the voucher, the name of the recipient and a greeting) about a gift voucher. Integer amounts can be deducted from the value of the gift voucher. When an instance of the `GiftVoucher` class is first created it is assigned an ID value; the ID values assigned to each new instance of the `GiftVoucher` class are in sequence.

For example, when the following code is executed with the completed `GiftVoucher` class:

```
GiftVoucher.setNextID(345);

GiftVoucher giftV1 = new GiftVoucher(250, "Richard");
giftV1.setGreeting("Happy Birthday");
System.out.println(giftV1.toString());
if (giftV1.hasEnoughFunds(130))
    giftV1.deductValue(130);
else
    System.out.println(giftV1.getVoucherID() + ": Not enough funds");
System.out.println();
System.out.println(giftV1.toString());
System.out.println();

GiftVoucher giftV2 = new GiftVoucher(50, "Jane");
System.out.println(giftV2.toString());
if (giftV2.hasEnoughFunds(51))
    giftV2.deductValue(51);
else
    System.out.println(giftV2.getVoucherID() + ": Not enough funds");
System.out.println();
giftV2.setGreeting("Congratulations!");
System.out.println(giftV2.toString());
```

the output is:

```
Voucher ID: 345
$250
TO: RICHARD
HAPPY BIRTHDAY
```

```
Voucher ID: 345
$120
TO: RICHARD
HAPPY BIRTHDAY
```

```
Voucher ID: 346
$50
TO: JANE
```

```
346: Not enough funds
```

```
Voucher ID: 346
$50
TO: JANE
CONGRATULATIONS!
```

SURNAME: FORENAMES:

```
/**
 * This class stores information about a gift voucher.
 *
 * @author Adriana Ferraro
 * @version 2003
 */
public class GiftVoucher {
    private static int nextID;    //next available ID

    private int value;            //value of the gift voucher
    private String greeting;      //greeting for the gift voucher
    private int voucherID;        //ID number of the gift voucher
    private String recipient;     //recipient of the gift voucher

    public GiftVoucher(int amount, String recipient) {
        
    }

    /**
     * Accessor method to set the
     * greeting for the voucher.
     *
     */
    public void setGreeting(String greet) {
        
    }

    /**
     * Accessor method used to obtain
     * the ID of the giftVoucher instance.
     *
     */
    public int getVoucherID() {
        
    }
}
```

CONTINUED

SURNAME: FORENAMES:

```
/**
 * Method which deducts money from the value of the voucher.
 * If the amount to be deducted is greater than the value of
 * the voucher then the value of the voucher is set to 0.
 */
*/
```

```
public void deductValue(int amt) {
```

```
}
```

```
/**
 * Method which returns true if the value of the voucher
 * is greater than or equal to the amount passed in as a parameter.
 */
*/
```

```
public boolean hasEnoughFunds(int amount) {
```

```
}
```

```
/**
 * Method to assign a value to the static variable, nextID.
 */
*/
```

```
public setNextID (int newNextID) {
```

```
}
```

SURNAME: FORENAMES:

```
/**
 * Method returns a String containing information about the
 * GiftVoucher instance. The greeting String is appended only if
 * it has a length greater than 0. The recipient and the greeting
 * are displayed in uppercase.
 */
public String toString() {
```

```
}
```

```
}
```

(12 marks)

SURNAME: FORENAMES:

Question 7 (8 marks)

The **transpose** of a matrix is defined as a matrix in which the rows and columns are interchanged. For example, here are a few matrices and their transposes,

matrix	1.0	2.0	3.0	transpose	1.0	4.0	7.0
	4.0	5.0	6.0		2.0	5.0	8.0
	7.0	8.0	9.0		3.0	6.0	9.0
matrix	8.0	9.0	7.0	transpose	8.0	3.0	
	3.0	4.0	5.0		9.0	4.0	
					7.0	5.0	
matrix	10.0			transpose	10.0	20.0	30.0
	20.0						
	30.0						

Design and write the Java code for a method with the following description and signature,

```
/**
 * Returns the transpose of a matrix.
 * @param data a two-dimensional array which represents a matrix
 * @return a reference to a new two-dimensional array that is the
 *         transpose of the array passed.
 */
public static double[][] transpose(double[][] data){
```

(8 marks)

SURNAME: FORENAMES:

Question 8 (4 Marks)

- (a) Name the 4 components of a **test case**.

- (b) Show an example of a command a test engineer would type to run a test case using a test class named `WidgetTest`. Assume that `WidgetTest` has a `main` method that reads test case data from standard input and that the test data is stored in a text file called `test.txt`

(4 marks)

SURNAME: FORENAMES:

Question 9 (10 Marks)

The Java util package has a class named `ArrayList`. Instances of this class store a list of objects of type `Object` in an internal array, and the class has methods such as `add`, `remove`, `contains` and others.

Each `ArrayList` instance has a *capacity*. The *capacity* is the length of the array used to store the elements in the list. It is always at least as large as the list size. As elements are added to an `ArrayList`, its *capacity* grows automatically (by doubling the existing capacity whenever the list becomes full). Part of the definition of the `ArrayList` class is shown below (a modified version that differs from the official Sun java.util version). Complete the definition for the `add` method shown on the facing page. Your answer should include a private helper method.

```
public class ArrayList {  
  
    /**  
     * The array into which the elements of the ArrayList are stored.  
     * The capacity of the ArrayList is the length of this array.  
     */  
    private Object[] elementData;  
  
    /**  
     * The size of the ArrayList (the number of elements it contains).  
     */  
    private int size;  
  
    /**  
     * Constructs an empty list with an initial capacity of ten.  
     */  
    public ArrayList() {  
        elementData = new Object[10];  
    }  
  
    /**  
     * Constructs an empty list with the specified initial capacity.  
     * The initial capacity must be greater or equal to zero  
     *  
     */  
    public ArrayList(int initialCapacity) {  
        elementData = new Object[initialCapacity];  
    }  
  
}
```

SURNAME: FORENAMES:

```
/**
 * Appends the specified element to the end of this list.
 *
 * @param obj element to be appended to this list.
 */
public void add(Object obj) {
```

}

```
//Include any helper methods here
```

*(10 marks)***CONTINUED**

SURNAME: FORENAMES:

Question 10 (10 Marks)

Here are the declarations of the instance variables, a constructor, a comparison method and a toString for a class named Address.

```
public class Address {

    /**
     * Instance variables
     */
    private int streetNumber;
    private String street;
    private String city;
    private int postalCode;
    private String country;

    /**
     * Constructor.
     */
    public Address (int streetNum, String streetName, String cityName,
                   int postCode, String countryName)

    /**
     * Compares a postal code with this Address object's postal code.
     *
     * @param  otherPostCode  another postal code
     * @return  true if this object's postal code is the same as the
     *          otherPostCode, else false.
     */
    public boolean equalsPostalCode(int otherPostCode)

    /**
     * Returns a String object representing this Address's values.
     * The String contains no newline characters.
     *
     * @return a string representation of all the Address object values
     */
    public String toString()
```

Assume the main method of an application class named MailingApp has declared an array of Address objects named customers and that **all** the array elements have been assigned references to Address objects. The application class also has a method named getAddressByPostalCode() defined as follows,

```
/**
 * Compares a postal code to the postal code of an array of Address
 * objects and returns a new array of Address objects whose postal
 * codes are the same as the postal code parameter.
 * @param  addrList  array of Address objects to search.
 *          All elements of the array are not null.
 * @param  code postal code to search for
 * @return  an array of Address objects containing the references
 *          to all the array objects in addrList whose postal code
 *          is the same as the code parameter.
 *          The array length may be greater than the number of
 *          matching addresses found.
 */
public static Address[] getAddressByPostalCode(
    Address[] addrList, int code)
```

CONTINUED

SURNAME: FORENAMES:

- (a) Write the code for the method `getAddressesByPostalCode`. Be sure to meet the specification given in the JavaDoc documentation and note especially the second sentence of the return description.

```
public static Address[] getAddressesByPostalCode( Address[] addrList,  
                                                int code){
```

- (b) Write a `while` loop that may be used in the `main` method to print the addresses in the array returned from a call to `getAddressesByPostalCode()`. The loop must print each `Address` on a new line and it must only print references to valid `Address` objects. Answer by completing the code below:

```
int postCode = getIntInput("Enter the postal code: ");  
Address[] addrByCode = getAddressesByPostalCode(customers, postCode);
```

(10 marks)

SURNAME: FORENAMES:

OVERFLOW PAGE

(If you have used this page, please indicate clearly under the relevant question that you have overflowed to this page)

SURNAME: FORENAMES:

APPENDIX: Useful classes, methods and constantsActionEvent and/or MouseEvent

```
public Object getSource()
public int getX()
public int getY()
```

ActionListener

```
public void actionPerformed(ActionEvent e)
```

Component

```
public void addActionListener(ActionListener listener)
public void addMouseListener(MouseListener listener)
public void addMouseMotionListener(MouseMotionListener listener)
public void paint(Graphics g)
public void repaint()
public void setBounds(int x, int y, int width, int height)
```

Graphics

```
public void fillRect(int x, int y, int width, int height)
public void drawRect(int x, int y, int width, int height)
public void drawOval(int x, int y, int width, int height)
public void drawLine(int x1, int y1, int x2, int y2)
public void setColor(Color c)
```

Math

```
public static int abs(int a)
public static int max(int a, int b)
public static int min(int a, int b)
public static double PI
public static double sqrt(double a)
```

MouseListener

```
public void mouseClicked(MouseEvent e)
public void mousePressed(MouseEvent e)
public void mouseReleased(MouseEvent e)
public void mouseEntered(MouseEvent e)
public void mouseExited(MouseEvent e)
```

MouseMotionListener

```
public void mouseDragged(MouseEvent e)
public void mouseMoved(MouseEvent e)
```

String

```
public char charAt(int index)
public String substring(int start, int end)
public boolean equals(String comparison)
public int length()
public String toUpperCase()
public String toLowerCase()
```

TextField

```
public String getText()
public void setText(String text)
```

SURNAME: FORENAMES:

ROUGH WORKING
(This page may be detached from the script
and used for working. It will not be marked)
