University of Cape Town
Department of Computer Science

Computer Science CSC116S

Test 2 - 8 September 2004

- Answer all questions.
- This test covers chapter 13 and 14 of the prescribed text book.
- Good luck !

Marks: 40

Time: 40 minutes

- Approximate marks per question are shown in brackets
- The use of calculators is permitted

	Surname	Initials
NAME:		

STUDENT NO: 　　　　 **COURSE CODE:** CSC

This paper consists of 5 questions and 6 pages (including this cover page).

Mark Allocation							
Quest	Marks	Internal	External	Quest	Marks	Internal	External
1	[14]			4	[4]		
2	[4]			5	[8]		
3	[10]						
	Total				**Total**		
					Grand Total		
					Final Mark		
Internal Examiner:				**External Examiner:**			

Question 1. **[14 marks]**

a) Describe the main difference between Stacks and Queues and list 5 methods in the java.util.Stack class, giving a brief description of each.

A stack is a LIFO structure
A Queue is FIFO (1 mark)

push(): put a new item on top of a stack (1 mark)
pop(): remove the topmost item it from the stack (1 mark)
peek(): get the value of the topmost item from the stack, but don't remove it from the stack (1 mark)
empty(): return true if the stack is empty (checks to see if the Stack is empty) (1 mark)
search(): tell how far a value is from the top of the stack, or -1 if the value is not in the stack (1 mark)

[6]

b) What is meant by Reverse Polish Notation and give an example?

With RPN you enter the operator after its operand.
Also called postfix notation.
(2 marks for explaining postfix or "operator being before operand")
Example is 68+ (operator before operand) (1 mark)

[3]

c) Show how the expression (8+4) x 2 can be converted to reverse polish notation using a stack. Show all the steps and not just the final solution.

```
                    (8+4) x 2

                Stack        Output

1.  Ignore (    [ ]
2.  Output 8    [ ]     8
3.  Push +      [+]     8
4.  Output 4    [+]     8 4
5.  Pop +       [ ]     8 4 +
6.  Push x      [x]     8 4 +
7.  Output 2    [x]     8 4 + 2
8.  Pop x       [+]     8 4 + 2 x

Final Solution: 84+2x
```

[5]

Question 2. **[4 marks]**

A queue class is given as follows:

```
public class Queue{
   Node front, rear; //front points to the beginning of the queue
                     //rear points to the end of the queue

   class Node        //inner class Node is the element of the queue
   {
        int value;
        Node next;
        private Node(int v)
        {
                value = v;
                next = null;
        }
   }
   public boolean isEmpty(); //check whether the queue is empty

   public Node peek(); //return a node from the front

   public Node dequeue(); //remove a node from the front

   public void enqueue(Node item); //add a new node at the end

   public int size(); //return the number of nodes in the queue

}
```

Implement the **enqueue** method:

```
 void enqueue(Node newNode) {
    if(newNode==null)
        throw new IllegalArgumentException
        ("Trying to enqueue null onto the queue");
    if( isEmpty())
        front = newNode;
    else
        rear.next = newNode;
    rear = newNode;
}
```

[4]

Question 3. [10 marks]

a) Write the definition of a Binary Tree [2]

Tree structure in which each node has 0, 1, or 2 children

b) Write the definition of Binary Search Tree [2]

It is a binary tree (don’t deduct for not stating this)
the key of every node in the left subtree is smaller than the key of this node
the key of every node in the right subtree is larger than, or equal to the key of this node

c) Consider the following binary tree:

```
               +---+
               | 3 |
               +---+
             /       \
           /           \
     +---+               +---+
     | 7 |               | 4 |
     +---+               +---+
    /     \                   \
   /       \                   \
+---+     +---+               +---+
| 2 |     | 6 |               | 0 |
+---+     +---+               +---+
         /     \              /    \
        /       \            /      \
    +---+     +---+      +---+     +---+
    | 1 |     | 9 |      | 5 |     | 8 |
    +---+     +---+      +---+     +---+
```

Fill in each of the traversals below:

i) Preorder Traversal [2]
3, 7, 2, 6, 1, 9, 4, 0, 5, 8

ii) Inorder Traversal [2]
2, 7, 1, 6, 9, 3, 4, 5, 0, 8

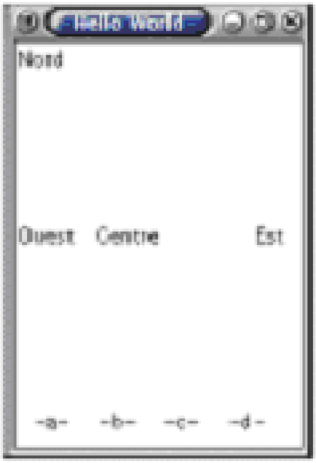
iii) Postorder Traversal [2]
2, 1, 9, 6, 7, 5, 8, 0, 4, 3

Question 4. **[4 marks]**

a) What are layout managers used for in java? [1]
Layout managers are used to control the size and position of components in containers

b) When MyFrame.java is compiled and run, a GUI is created. Draw the resulting GUI. [3]

```
import java.awt.*;
public class MyFrame extends Frame {
      public MyFrame(String title) {
           super(title);
           setLayout(new BorderLayout());
           add(new Label("Nord"), BorderLayout.NORTH);
           add(new Label("Est"),BorderLayout.EAST);
           add(new Label("Ouest"),BorderLayout.WEST);
           add(new Label("Centre"),BorderLayout.CENTER);
           Panel p = new Panel();
           p.setLayout(new FlowLayout());
           p.add(new Label("-a-"));
           p.add(new Label("-b-"));
           p.add(new Label("-c-"));
           p.add(new Label("-d-"));
           p.add(new Label("-e-"));
           add(p,BorderLayout.SOUTH);
           setSize(200,300);
           setVisible(true);
       }
 }
```



1 mark for getting abcde bottom aligned and flowing from left to right
0.5 for each of the other 4 labels

Question 5. **[8 marks]**

a) Fill in the following blanks using the java code below.

i) class SquareActionListener implements **ActionListener** [2]

ii) public void **actionPerformed** (ActionEvent e) [2]

iii) **setLayout** (new GridLayout(1,2)); [2]

```
import java.awt.*;
import java.awt.event.*;
public class Square extends Frame {
     Button button = new Button("Square");
     TextField input = new TextField();

     public Square() {
         super("Square GUI");
         _____________ (new GridLayout(1,2));
         add(button);
         add(input);
         button.addActionListener(new
         SquareActionListener(this));
         pack();
         show();
     }
     protected void square() {
         int v = Integer.parseInt(input.getText());
         input.setText(Integer.toString(v*v));
     }
}
class SquareActionListener implements ___________ {
    private Square appl;
    SquareActionListener(Square appl) {
        this.appl = appl;
    }

    public void __________(ActionEvent e) {
        appl.square();
    }
}
```

b) Fill in the blank: JButton, JTextField and JComboBox are examples of **atomic components** . [1]

c) What does the pack() method of the JFrame class do? [1]
Resizes the frame so that all the atomic components fit into it