RHODES UNIVERSITY
COMPUTER SCIENCE DEPARTMENT

EXAMINATIONS: JUNE 2007

COMPUTER SCIENCE 201

Examiner: Dr KL Bradshaw
Prof GC Wells
Prof PD Terry

Total Time: 180 Minutes
Total Marks: 180 Marks

External Examiner: Prof S Berman

Instructions to candidates:

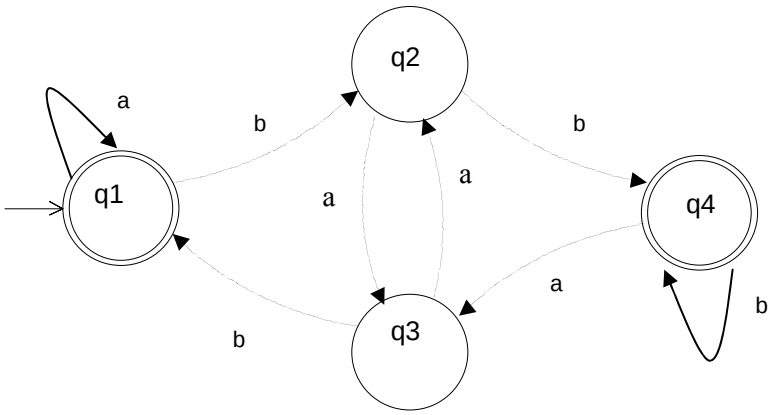
a) Please answer all questions on the question paper.
b) There are 3 Sections, 23 Pages and 16 Questions, ***PLEASE MAKE SURE THAT YOU HAVE A COMPLETE PAPER.***
c) Calculators may **NOT** be used in Section C of this paper.
d) Section C has free information which is attached to the end of the paper. Pages 21 – 23.

PLEASE NOTE – most of the blank space required for the answers has been removed in this version of the exam paper (10 pages).

SECTION A:
THEORY OF COMPUTING 35 MARKS

Question 1 **(6 Marks)**

Given below is the transition diagram for a Finite State Automaton.



a) Is this FSA deterministic or not? Give a reason for your answer. [2]

b) What is the set of accept states? [1]

c) What sequence of states does the machine go through on input aabaab? [1]

d) Give a string that is rejected by the FSA. Your string must use the alphabet of the FSA and have length ≥ 5. [2]

Question 2 **(11 Marks)**

a) A grammar is said to be ambiguous if there are two ways of deriving a given string. The string **001101** has two possible derivations in the ambiguous grammar given below. Give both these derivations. [5]

```
<S> → 0 <A>
<S> → 1 <B>
<A> → 0 <A> <A>
<A> → 1 <S>
<A> → 1
<B> → 1 <B> <B>
<B> → 0 <S>
<B> → 0
```

Derivation 1 Derivation 2

b) Design the least powerful automaton that will accept strings from the following language: [6]

$\{ 1^k 0^j 1^j 0^k$ where j, k > 0$\}$

Question 3 **(12 Marks)**

a) List the components of a Turing Machine (TM) and state how each contributes to the workings of the TM. [6]

b) Explain in plain English (with or without the aid of diagrams), the steps a TM, designed to detect palindromes, might take in rejecting the following string:
101001 [6]

Question 4 **(6 Marks)**

Prove by contradiction that it is impossible to write a program that determines whether all programs will halt for a given input.

SECTION B: ADVANCED PROGRAMMING 90 MARKS

Question 5 **(8 Marks)**

a) _____ is a tool used to automatically create documentation for Java programs. [2]

b) What is an *iterator*? [2]

c) What is the definition of a *graph*? [2]

d) Given a choice between an $O(n^2)$ algorithm and an $O(2^n)$ algorithm, which would you choose? [2]

Question 6 **(4 Marks)**

The Fibonacci series is easily generated recursively. The n^{th} Fibonacci number (f_n) is defined to be the sum of the previous two Fibonacci numbers ($f_{n-1} + f_{n-2}$). The first two Fibonacci numbers (f_1 and f_2) are defined to be 1. The first few values in the series are: 1, 1, 2, 3, 5, 8, 13, etc. More formally, we can define the Fibonacci series:

$f_n = 1$ If $n = 1$ or $n = 2$
$f_n = f_{n-1} + f_{n-2}$ If $n > 2$

Write a recursive Java method to calculate the n^{th} number in the Fibonacci series.

```
public long fibonacci (int n)
    {

    } // fibonacci
```

Question 7 **(12 Marks)**

Complete the inner `DequeIterator` class below so that it can be used as an iterator for the `Deque` class. The iterator should work from left to right through all the data contained in the deque.

```
public class Deque<T>
  { private class DequeNode
      { public T data;
        public DequeNode lt; // Pointer to left neighbour
        public DequeNode rt; // Pointer to right neighbour
      } // inner class DequeNode

    private class DequeIterator implements Iterator<T>
      {


        public DequeIterator ()
          {

          } // constructor

        public T get ()
        // Get the current item
          {

          } // get

        public void next ()
        // Move to the next item
          {

          } // next

        public boolean atEnd ()
        // Tell whether there are any more items
          {

          } // atEnd

      } // inner class DequeIterator

    private DequeNode header; // Reference to the header node.

    public Deque ()
    //  Create an empty deque.
      { header = new DequeNode(); // Create header node
        header.lt = header;
        header.rt = header;
      } // Constructor
```

```
  public void addLeft (T item)
  // Add an item to the left end of the deque.
    { DequeNode newNode = new DequeNode();
      newNode.data = item;
      newNode.rt = header.rt;
      newNode.lt = header;
      header.rt.lt = newNode;
      header.rt = newNode;
    } // addLeft

  public void addRight (T item)
  // Add an item to the right end of the deque.
    { DequeNode newNode = new DequeNode();
      newNode.data = item;
      newNode.lt = header.lt;
      newNode.rt = header;
      header.lt.rt = newNode;
      header.lt = newNode;
    } // addRight

  public T removeLeft ()
  // Remove an item from the left end of the deque.
    { assert header.rt != header : "Deque is not empty";
      DequeNode tmpPtr = header.rt;
      T tmpData = tmpPtr.data;
      header.rt = tmpPtr.rt;
      tmpPtr.rt.lt = header;
      return tmpData;
    } // removeLeft

  public T removeRight ()
  // Remove an item from the right end of the deque.
    { assert header.lt != header : "Deque is not empty";
      DequeNode tmpPtr = header.lt;
      T tmpData = tmpPtr.data;
      header.lt = tmpPtr.lt;
      tmpPtr.lt.rt = header;
      return tmpData;
    } // removeRight

  public boolean isEmpty ()
  // Tell whether the deque is empty.
    { return header.lt == header;
    } // isEmpty

  public Iterator<T> getIterator ()
  // Obtain an iterator for this deque.
    { return new DequeIterator();
    } // getIterator

} // class Deque
```

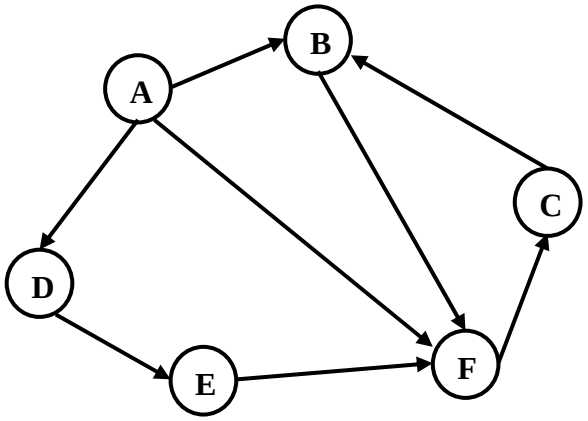
Question 8 **(16 Marks)**

Implement the `add`, `remove`, `head` and `isEmpty` methods for the following `ArrayQueue` class. Use a circular-array technique.

```
public class ArrayQueue<T> implements Queue<T>
  { private T[] data; // The array of data.
    private int hd; // Index of the item at the head of queue.
    private int tl; // Index of the item at the tail of queue.

    public ArrayQueue (int initSize)
     // Create an empty queue, with a given capacity.
      { data = (T[])new Object[initSize];
        hd = tl = -1;
      } // Constructor

    public ArrayQueue ()
    // Create an empty queue, with a default capacity of
    // 100 elements.
      { this(100);
      } // Constructor

    public void add (T item)
    // Add an item to the tail of a queue.
      {

      } // add

    public T remove ()
    // Remove an item from the head of a queue.
      {

      } // remove

    public T head ()
    // Return a copy of the item at the front of a queue,
    // without removing it.
      {

      } // head

    public boolean isEmpty ()
    // Tell whether the queue is empty.
      {

      } // isEmpty

  } // class ArrayQueue
```

Question 9 **(10 Marks)**

Show the adjacency matrix for the following directed graph:



	A	B	C	D	E	F
A						
B						
C						
D						
E						
F						

Question 10 **(12 Marks)**

Describe how internal and external hash tables handle the problem of collisions, illustrating your description with an example. What are the advantages and disadvantages of each approach?

Question 11 **(14 Marks)**

a) Draw the binary search tree that is equivalent to the way in which the binary search algorithm searches the following list of numbers. [10]

0	1	2	3	4	5	6	7	8	9
2	5	7	9	14	20	29	33	38	43

b) Use arrows on your tree in part (a) to show how the tree would be searched for the value 17, which is not in the list. [2]

c) What is the complexity of the binary search? [2]

Question 12 **(14 Marks)**

a) Describe the Selection Sort algorithm (you do not need to write Java code, but may do so if you prefer). [10]

b) What is the complexity of the Selection Sort in the average, best and worst cases, considering both the number of comparisons and data movement? [4]

SECTION C: ARCHITECTURE 55 MARKS

Question 13 (Number representation) **(12 Marks)**

(Calculators may not be used! Show your workings clearly.)

a) An easy one to start you off. Suppose the number "thirty", when expressed in base *x*, is represented as 132_x. What is the value of *x*? [3]

b) A computer geek has modified his motor car so that the odometer (the dial that shows how far the car has travelled) gives the distance in octal notation, rather than the more usual decimal notation. Currently the dial reads 25516. How many kilometres has the vehicle travelled (give your answer in words, for example "ten thousand and forty three") [3]

c) He takes his girlfriend on a joyride to Port Elizabeth. One hundred and nineteen kilometres outside of Grahamstown they are caught in a speed trap. What does the odometer read at this stage? [2]

d) The geek is considering modifying his car yet again so that the odometer will display the distance in hexadecimal notation rather than octal. What would the original reading of 25516 look like in that case? [2]

e) Another geek thinks it would be cool to use dotted decimal notation instead. As you should know, this notation is typically used for "large" values of a base, like 256 - but what would the dotted decimal notation representation look like if a base of 16 were used? [2]

Question 14 (Boolean algebra and combinational circuits) **(17 Marks)**

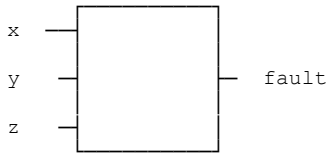
a) Boolean algebra is usually defined on the fundamental basic operators (' . and +), otherwise denoted by NOT, AND and OR. Other "gates" may be expressed in terms of these; one of these, the XOR operator is often regarded as "fundamental" as well.

Define the XOR operator in terms of the AND, OR and NOT operators and give a truth table for it. [2]

b) How do you form the "dual" of a Boolean expression? Illustrate your solution by forming (and then simplifying) the dual of the expression given in (a). [3]

c) The inverse or complement of the XOR operator is known as the EQUIV operator. Complement the expression given in (a), and hence or otherwise show that the dual of the XOR operator is the same as the EQUIV operator. [2]

d) A piece of equipment has three fault detectors x, y and z, and these are to be connected to an alarm unit through a logic circuit. The presence of a single fault is deemed to be unimportant, but if more than one fault is detected the alarm should go off.



Construct a truth table for the circuit, and write down a disjunctive normal form of the expression that defines fault = f(x, y, z) [4]

e) Hence or otherwise show how this expression can be simplified to [4]

fault = x . y + y . z + z . x

f) The expression above would lead to a combinational circuit employing five gates. Show that it is possible to find at least one equivalent circuit with only four gates.[2]

Question 15 (Assembler programming) (7 Marks)

Consider the following (typically uncommented!) program for the toy assembler/machine used in this course:

```
      BEG
      LDI   X
LOOP  OTI
      ADI   LOOP
      BPZ   LOOP
      HLT
X     EQU   'x'
      END
```

a) Suppose this program were assembled and stored in memory. Show the contents of memory immediately after this process is completed. Using the free information, express all values in HEXADECIMAL. [3]

0	1	2	3	4	5	6	7	8	9

b) What would be the exact output from this program if it were then executed? Explain your reasoning. [4]

Question 16 (Assembler programming) (19 Marks)

Two students have been revising for the examinations by working through the exercises in the course handouts. They have come across one which reads:

> Read a list of positive numbers (terminated by a negative number that is not part of the list) and print out only the unique numbers which appear in the list.

One student argues that this should be solved with an algorithm expressed essentially as

```
boolean [] seen = new boolean[max + 1];
x = max;
do {                // none seen yet
  seen[x] = false;
  x--;
} while (x >= 0);
while (true) {
  i = IO.readInt();
  if (i < 0) break;
  if ( ! seen[i] ) {
    IO.writeInt(i); seen[i] = true;
  }
}
System.exit(0);
```

The other student argues that the problem really calls for an algorithm like

```
boolean [] seen = new boolean[max + 1];
x = max;
do {                // none seen yet
  seen[x] = false;
  x--;
} while (x >= 0);
while (true) {
  i = IO.readInt();
  if (i < 0) break;
  seen[i] = true;
}
i = 0;
do {
  if (seen[i]) IO.write(i);
  i++;
} while (i <= max);
System.exit(0);
```

a) Are these suggestions, in fact, equivalent? If not, what is the difference between them? [3]

b) The students decide to code their solutions into assembler code for the little byte machine simulator used in their course. They realise that there must be a limit on the value they can use for `max`. Suggest how they can determine this limit. [2]

. c) Give the assembler code for a solution that you think solves the problem. [14]

END OF THE EXAMINATION