Please fill in your Student Number and Name.

Student Number : _____

Name:

_____

_____

Student Number:

_____

University of Cape Town ~ Department of Computer Science

Computer Science 3003S ~ 2009

Test 1

Question	Max	Mark	Internal	External
1	12			
2	10			
3	8			
4	15			
TOTAL	**40**			

Marks : 40

Time : 45 minutes

Instructions:

a) Answer all questions.

b) Write your answers in the space provided.

c) Show all calculations where applicable.

Question 1 – Lexical Analysis [12]

a) What is a Deterministic Finite Automaton (DFA)? [2]

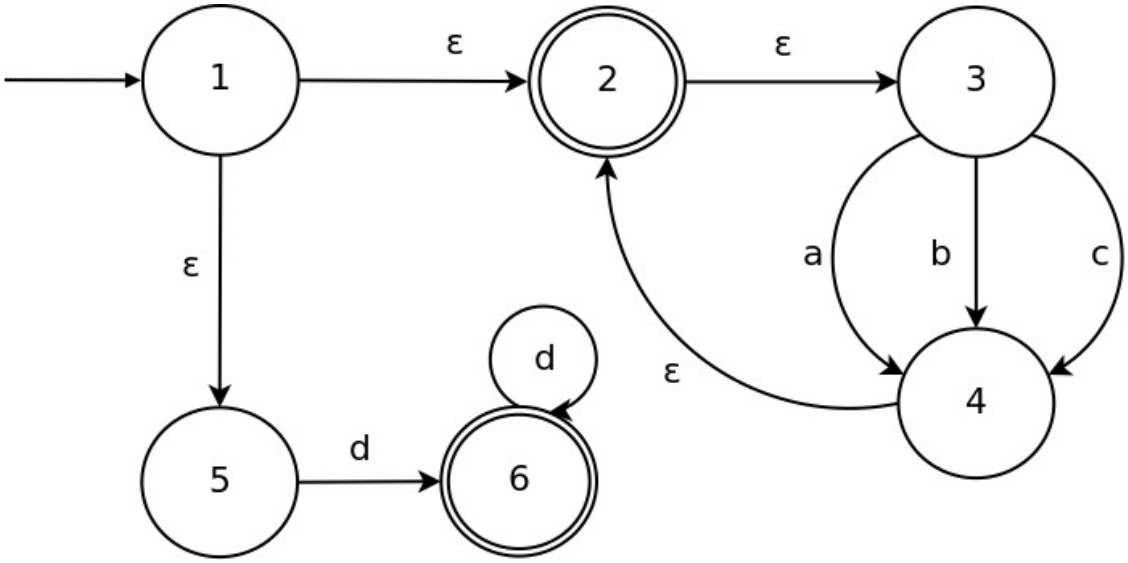
A finite state machine [1]. At any given state, for any given symbol, there exists at most 1 transition [1].

b) Consider the following regular expression and describe the strings it matches: [3]

```
[ 0 – 9 ] ( [ a – z ] | [ 0 – 9 ] ) *
```

Strings starting with a digit [1], followed by 0 or more [1] letters or digits [1]

c) Convert the Non-Deterministic Finite Automaton (NFA) below to a Deterministic Finite Automaton (DFA). Remember to indicate the start and final state(s). [7]

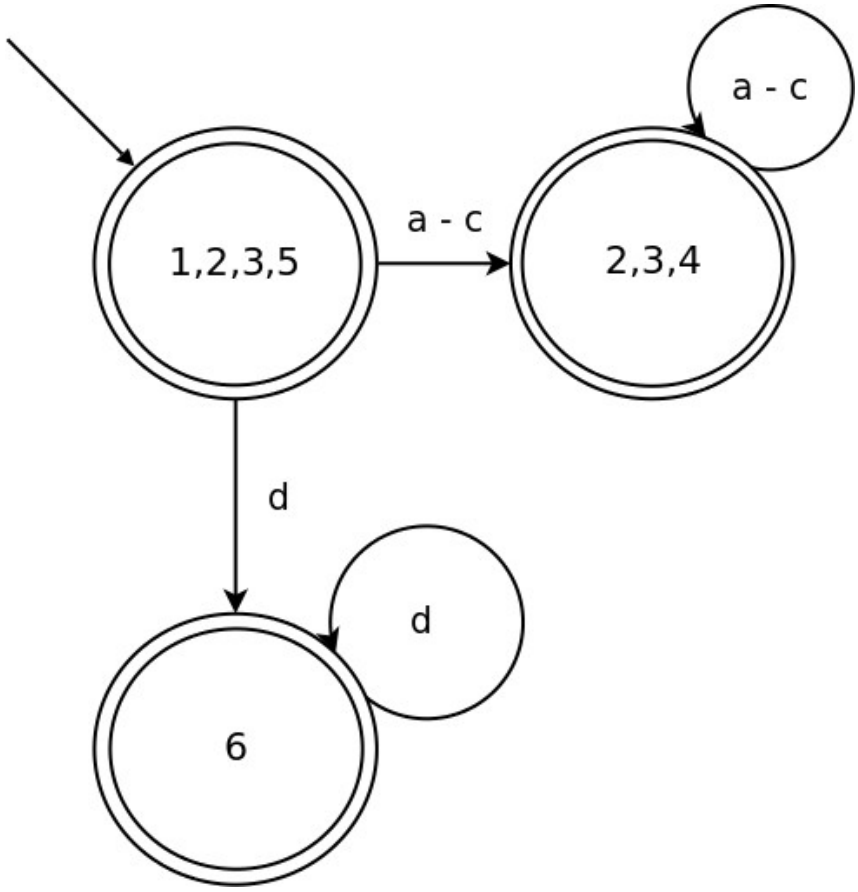


Solution DFA shown below

½ Mark for correct state (labels may differ), maximum 1½ marks

½ Mark for showing that state is final, maximum 1½ marks

1 Mark per correctly labelled transition (if student shows separate edges for a, b and c then only 1 mark is given for all of them), maximum 4 marks



Question 2 – Parsing [10]

a) Consider the following grammar and complete the table below by calculating its nullable, FIRST and FOLLOW sets ($ should be treated as a terminal): [5]

```
S → E$
E → aFc
E → aF
F → b
F →
```

	nullable	FIRST	FOLLOW
S			
E			
F			

Solution table shown below

1 Mark for correct nullable column

1 Mark for correct FIRST column

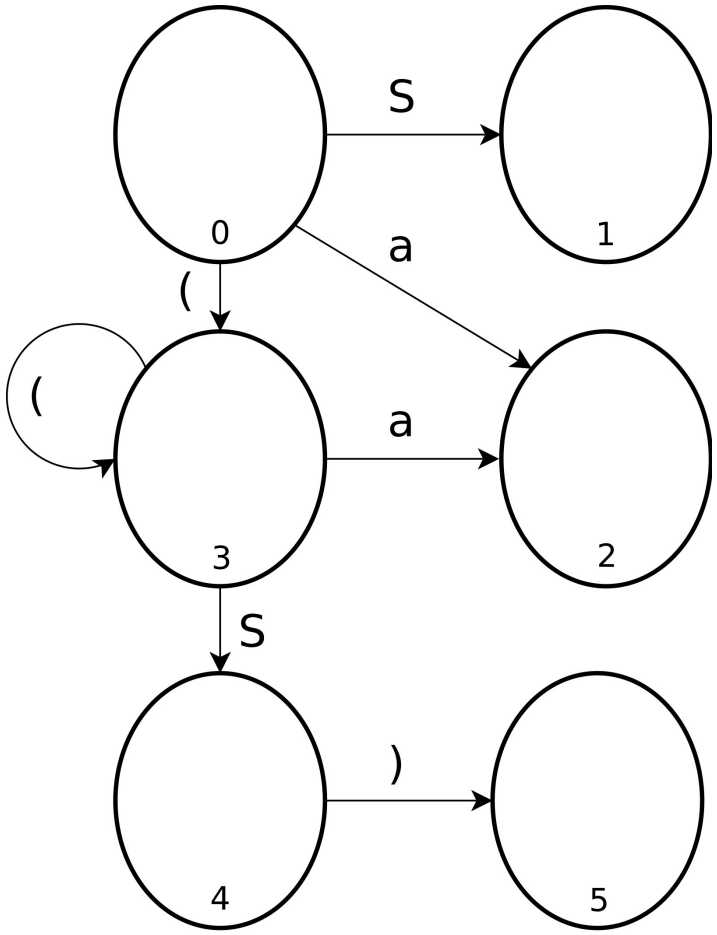
1 Mark for correct FOLLOW entry for E

2 Marks for correct FOLLOW entry for F (ignore comas)

	nullable	*FIRST*	*FOLLOW*
S	*FALSE*	*a*	
E	*FALSE*	*a*	*$*
F	*TRUE*	*b*	*c, $*

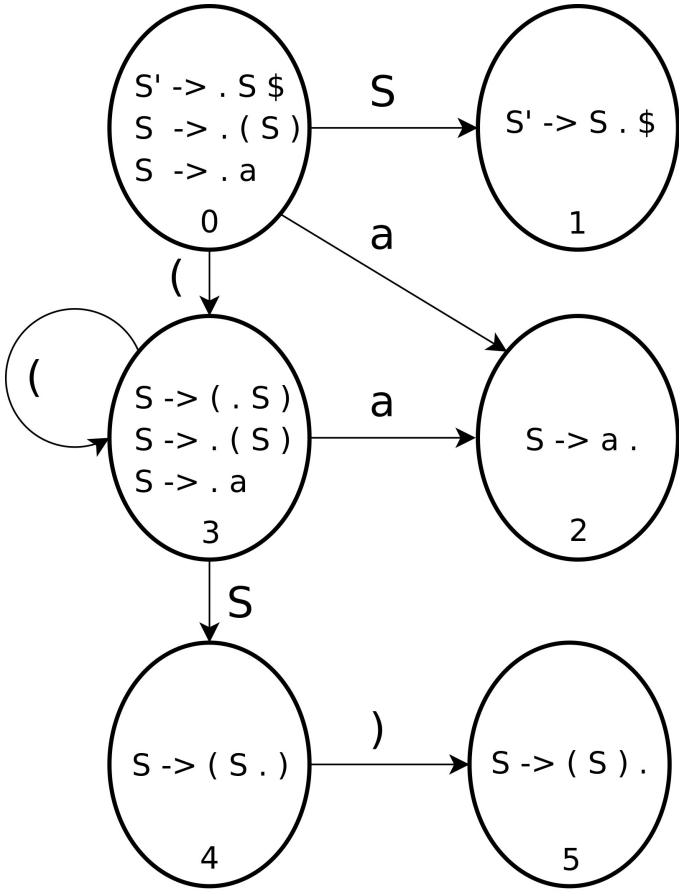
b) Consider the following grammar and complete the LR(0) automaton below: [5]

```
S' → S$
S  → (S)
S  → a
```



Solution automaton on the right

½ Mark for each correct LR(0) item (production with a dot), max 10 items => 5 marks



Question 3 – Error Recovery [8]

There are two main classes of error recovery techniques:

- Local Error Recovery
- Global Error Recovery

a) What is a difference between these two classes? [2]

Global Error Recovery adjusts the input before the point where the error was detected [1]
Local Error Recovery adjusts the input stream at the point of error (or immediately after) [1]

b) What adjustments can be made to the input stream to enable parsing to continue? [3]

Tokens can be deleted [1], inserted [1] or replaced/substituted [1]

Burke-Fisher is an example of an error recovery technique.

c) Given a token window size of 5, how many alterations can be made to the input stream to enable parsing to continue? [1]

One [1]

d) For which edge case(s) does this technique fail (assuming there are no semantic actions that affect lexical analysis) and why? [2]

Burke-Fisher works by single token manipulation of the input stream [1], thus it fails if there is no single token adjustment that will allow parsing to continue [1].

i.e. 1 mark for stating single token manipulation and 1 mark for linking it to edge case

Question 4 – Semantic Analysis [15]

Semantic analysis follows syntactic analysis and is the process of determining context-sensitive information and thereafter generating code. Answer the following questions related to semantic analysis.

a) The starting point in semantic analysis is usually an Abstract Syntax Tree. What is the difference between a Concrete Syntax Tree and an Abstract Syntax Tree? [2]

Concrete syntax tree contains all tokens from a grammar as derived from parsing. [1]

Abstract syntax trees contain equivalent semantic information in concise tree structures. [1]

b) Symbols tables are used to perform context-sensitive checking. Name 2 types of information and 2 distinct attributes that may be stored in a symbol table. [2]

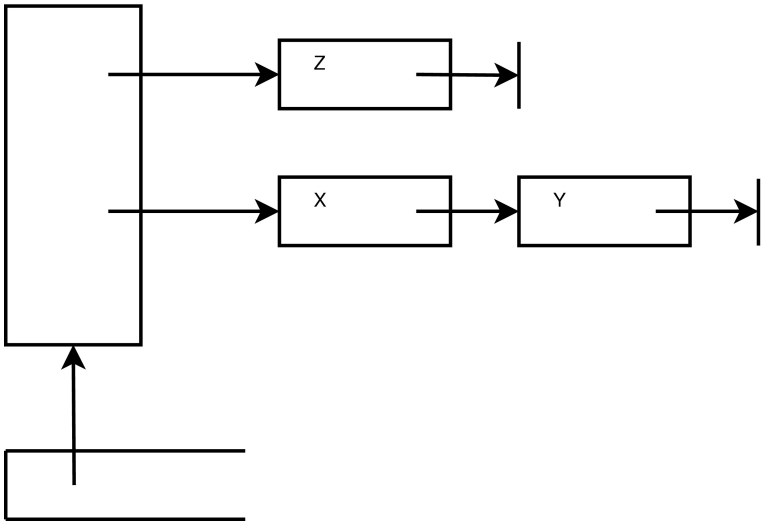
information: variable definitions, type definitions, subprogram names, etc. [½ x 2]

types: names, type, size, lexical level, etc. [½ x 2]

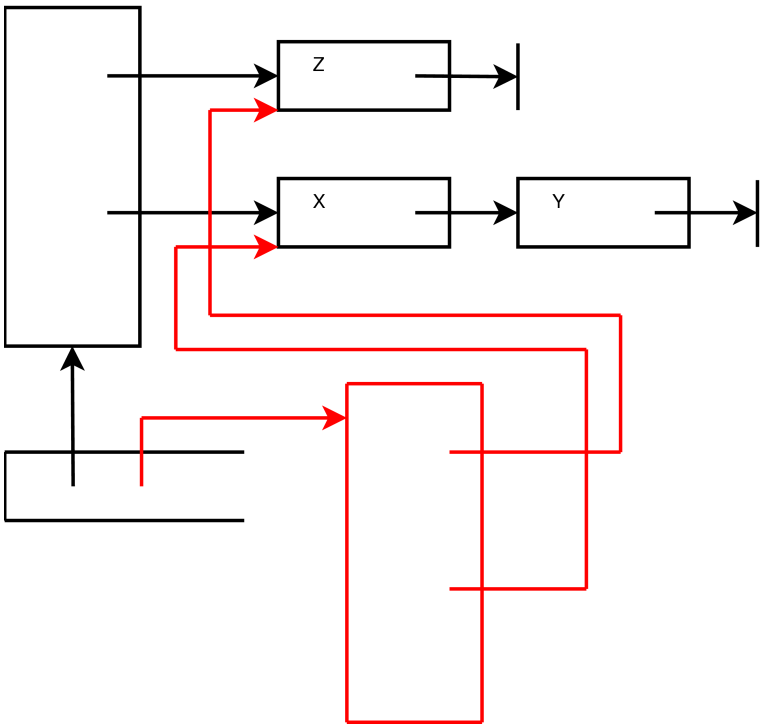
c) Scope must be taken into account in a symbol table. How does static scope differ from dynamic scope? [2]

static scope is determined by lexical nesting of subprograms [1] – dynamic scope is determined by call sequence [1]

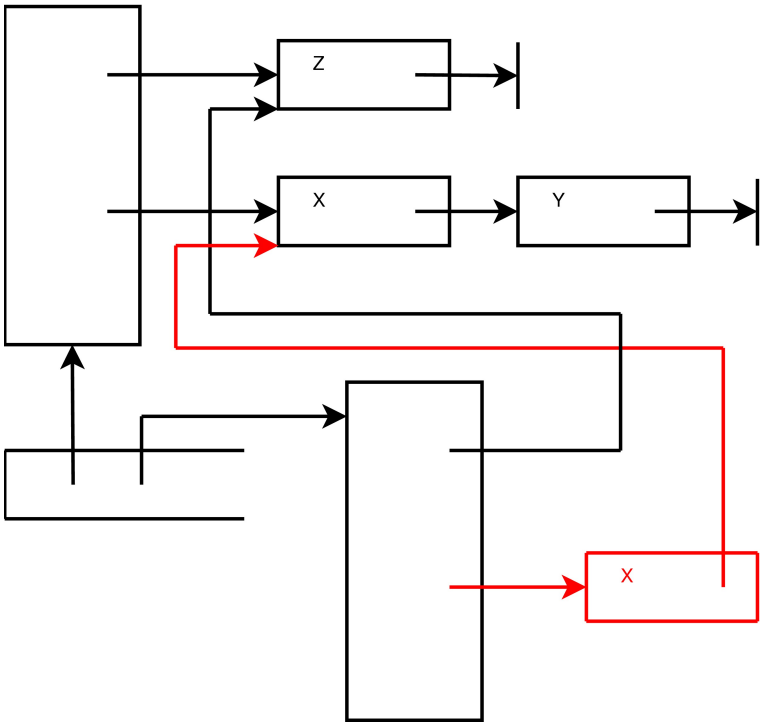
d) Show diagrammatically how the following **functional hash table** implementation of a symbol table is modified (A) when a new scope is opened and (B) when a new definition of X is added in that scope. [4]


Z
X
Y

[2]



[2]



e) Briefly explain 2 techniques to resolve non-local references at runtime in statically-scoped languages. [2]

follow static links to get to lexically-nested parents' activation records. [1]

use a display of links to subprograms at different lexical levels of nesting [1]

f) The following DDD assembler source code listing corresponds to the unoptimised C function provided. In the context of subprogram invocation mechanics, circle and indicate which memory address or register use corresponds to: (A) a local variable; (B) the frame pointer; and (C) the return value. [3]

```
        push   %ebp
        mov    %esp,%ebp
        sub    $0x10,%esp
        mov    0x8(%ebp),%eax
        add    $0x1,%eax
        mov    %eax,-0x4(%ebp)
        mov    -0x4(%ebp),%eax
        leave
        ret

int testsub ( int x )
{
   int a = x+1;
```

```
    return a;
}
```

A – any -0x4(ebp) [1]

B – any ebp [1]

C – eax in last mov statement [1]