University of Cape Town ~ Department of Computer Science

Computer Science 3003S ~ 2009

November Exam

Marks : 100

Time : 180 minutes

Instructions:

a) Answer ALL questions in Sections A and C.

b) Answer 3 questions in Section B and 3 questions in Section D.

c) Show all calculations where applicable.

Section A : COMPILERS [Answer questions 1 and 2 – both compulsory]

Question 1 : You must answer this question. [10]

a) Explain the difference between a grammar and the language it generates. [2]

A grammar is a finite set of rules/recipe [½ mark] for generating/recognizing all valid strings in a language [½ mark], whereas a language is a possibly infinite set of strings [½ mark] made of symbols from fixed alphabet. [½ mark]

b) When is a context free grammar ambiguous? [2]

If there are multiple ways [1 mark] of generating the same string [1 mark].

If a string [1 mark] has more than one parse tree [1 mark].

If a string [1 mark] has more than one leftmost/rightmost derivation [1 mark]. (Different leftmost and rightmost derivation does not imply ambiguity).

Maximum of 2 marks.

c) Describe in detail how a lexical analyser can be automatically constructed from a list of regular expressions and used to match tokens in an input stream. Your description should mention how to disambiguate rules and how the longest matching substring can be found. [6]

Discussion should mention:

Regular Expressions can be converted to DFA. [1 mark]

DFA can be converted to NFA. [1mark]

Need to disambiguate final states using rule priority [1 mark]

DFA can be interpreted/executed directly by computer [1 mark]

How to find longest matching substring [1 mark]

What token corresponds to matched substring [1 mark]

Example Answer:

The list of regular expressions can be algorithmically converted into a NFA [1 mark] and then into a DFA [1 mark].

If a DFA state consists of multiple final NFA states then the token corresponding to the regular expression which appears first in the list (has the highest rule priority) is accepted [1 mark].

The DFA can be converted to a matrix of rules for state transitions which can be directly interpreted by a computer [1 mark].

The matrix can then be applied to an input stream, where every time a final state is reached, the state number and input position are recorded [1 mark].

When there is no more input or no transition for the current input symbol, the token accepted corresponds to the token accepted by the last final state and consists of the input from the beginning of the input stream to the input position previously recorded [1].

Question 2: You must answer this question. [10]

Compilers that generate code for mobile devices must pay particular attention to the restrictions of mobile devices. Answer the following questions in the context of a mobile device with very little memory to store both instructions and data.

a) What is the purpose of the compiler? [1]

convert from high level language to low level language

b) Describe one advantage and one disadvantage of separating the compiler front-end from the compiler back-end. [2]

ad: easier to retarget compiler to new machine [1], easier to apply optimisations to IR [1], easier to build compiler for new language [1]

disad: more work [1] slower compilation [1]

c) Which peephole optimisations from the following list will not typically result in larger code size?

Constant propagation; Constant folding; Inlining; Common subexpression elimination; Strength reduction; Loop unrolling [3]

Constant propagation; Constant folding; Common subexpression elimination; Strength reduction [-1/2 for each missing or wrongly included item]

d) In the dynamic programming instruction selection algorithm, how would you ensure that the generated code is as small as possible? [2]

assign each tile cost based on the size of the resulting instruction

e) When allocating registers, MOVE-related nodes potentially result in smaller code. How does this happen? [2]

If the nodes can be coalesced, after colouring of the graph both original nodes have the same register. This results in a MOVE with the same source and destination, which can be safely deleted from the code.

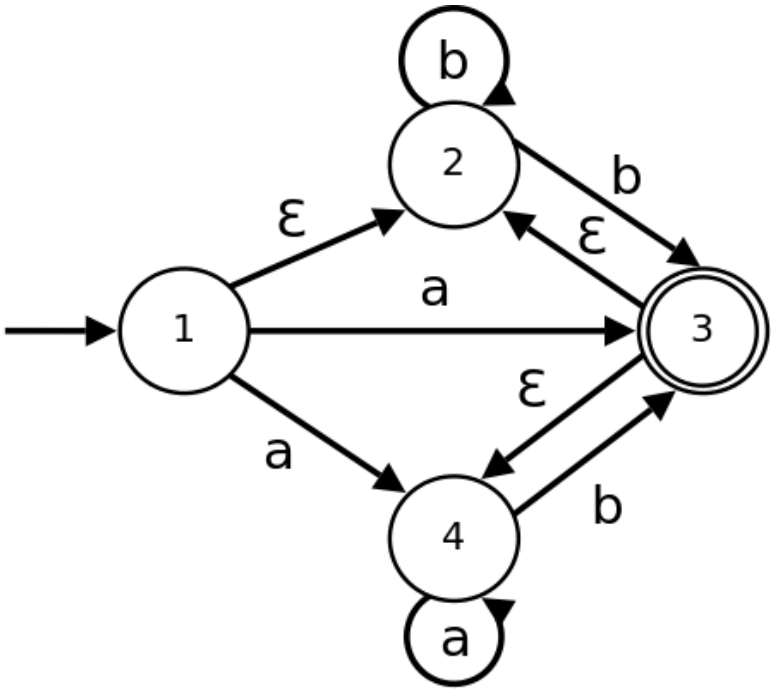
Section B COMPILERS [Answer any 3 questions ONLY]

Question 3 – Lexical Analysis [10]

a) For the alphabet {a,b}, find a regular expression that matches all strings of even length, starting with "a". [2]

a(a|b)((a|b)(a|b)) 1 Mark for first part a(a|b), 1 Mark for second part ((a|b)(a|b))**

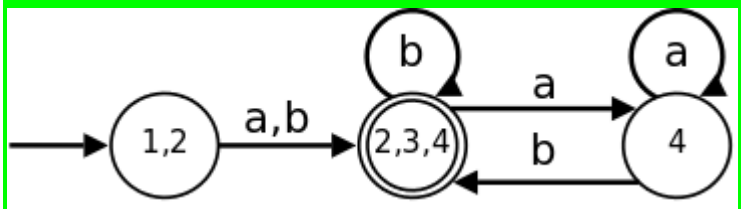
b) Consider the Non-Deterministic Finite Automaton (NFA) below:



i. Convert it to a Deterministic Finite Automaton (DFA). Remember to indicate the start and final state(s). [6]

1 Mark for each correct state (must correctly identify start and final state, to receive marks for those states). Maximum 3 Marks.

1 Mark for each state where all originating edges are correct. Maximum 3 Marks.



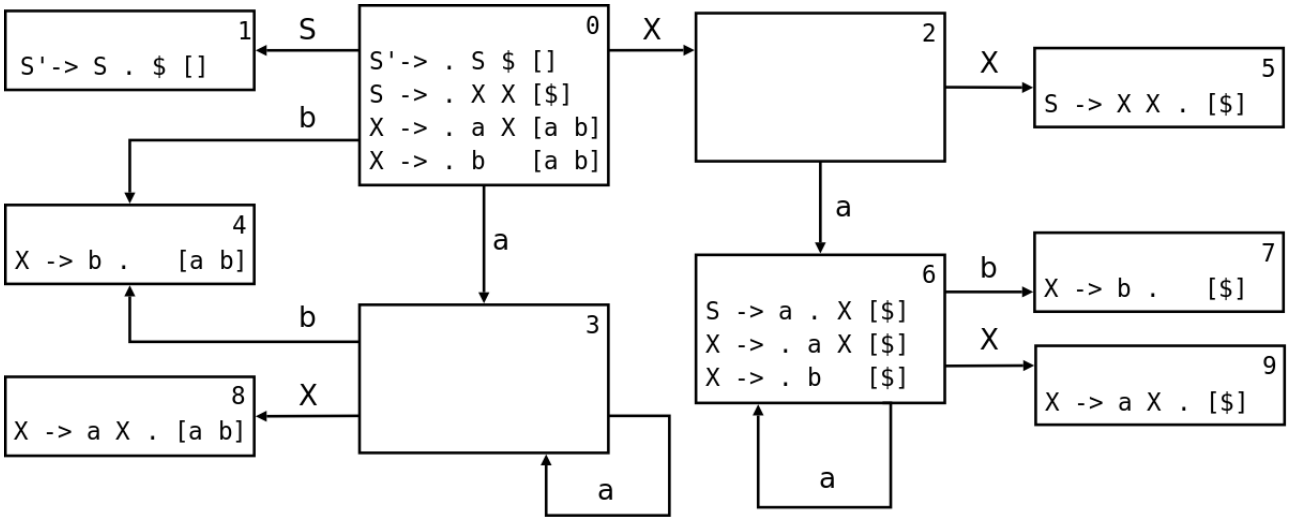
ii. Two strings of length 3 that are clearly acceptable are “aab” and “bbb”. Find all other strings of length 3 that are also acceptable. [2]

abb [1 mark], bab [1 mark]

Question 4 – Parsing [10]

Consider the following grammar and LR(1) automaton and answer the questions below:

```
0: S' → S $
1: S → X X
2: X → a X
3: X → b
```



a) Complete State 2 of the LR(1) Automaton. [3]

S → X . X [$] [1 mark]

X → . a X [$] [1 mark]

X → . b [$] [1 mark]

b) Complete State 3 of the LR(1) Automaton. [3]

X → a . X [a b] [1 mark]

X → . a X [a b] [1 mark]

X → . b [a b] [1 mark]

c) Is the automaton an LALR(1) automaton? Motivate your answer. [2]

No, Automaton has states which only differ on look-ahead/duplicate states [1 mark], e.g. State 3 and 6 [1 mark] or State 4 and 7 [1 mark] or State 8 and 9 [1 mark]. (Maximum 2 Marks)

d) Complete the LR(1) parsing table entries for State 0 and State 8. Use the template provided below when answering this question. [2]

	a	b	$	S	X
0					
8					

For state 0: (max 1 mark)

½ mark for correct shift actions

½ mark for correct goto actions

For state 8: (max 1 mark)

½ mark per correct reduce action

	a	b	$	S	X
0	s3	s4		g1	g2
8	r2	r2			

Question 5: Basic Blocks and Canonicalisation [10]

a) You have been given the following list of canonicalisation rules:

```
ESEQ(s1, ESEQ(s2, e))    => ESEQ(SEQ(s1,s2), e)
BINOP(op,ESEQ(s,e1),e2)  => ESEQ(s, BINOP(op, e1, e2))
MEM(ESEQ(s, e1))         => ESEQ(s, MEM(e1))
JUMP(ESEQ(s, e1))        => SEQ(s, JUMP(e1))
MOVE(ESEQ(s, e1), e2)    => SEQ(s, MOVE(e1, e2))
MOVE(e1, ESEQ(s, e2))    => SEQ(MOVE(TEMP t, e1), SEQ(s, MOVE (TEMP t, e2)))
```

Transform this IR tree as much as possible, using the above rules. Show each rule being applied as a separate step.

MOVE (ESEQ, (s1, ESEQ (s2, e1)), MEM (ESEQ (s3, e2))) [4]

MOVE (ESEQ, (s1, ESEQ (s2, e1)), ESEQ (s3, MEM (e2)))

MOVE (ESEQ (SEQ (s1, s2), e1), ESEQ (s3, MEM (e2)))

SE Q (SEQ (s1, s2), MOVE (e1, ESEQ (s3, MEM (e2))))

SE Q (SEQ (s1, s2), SE Q (MOVE (TEMP t, e1), SEQ (s3, MOVE (TEMP t, MEM (e2)))))

b) What is a basic block? [2]

sequence of instructions starting with label, ending with jump, no intervening labels/jumps

c) Perform basic block and trace analysis on the following code. Clearly show (i) the initial set of basic blocks; (ii) the set of traces; and (iii) the final optimised code. [4]

```
Start: ----------
       jump E
B:     ----------
       jump F
C:     ----------
       jump B
D:     ----------
       jump C
E:     ----------
       jump D
F:     ----------
```

Assume the -------s refer to sequences of statements.

(I) [1]

Start: ----------

jump E

B: ----------

jump F

C: ----------

jump B

D: ----------

jump C

E: ----------

jump D

F: ----------

jump Done

(ii) [1]

Start: ----------

jump E

E: ----------

jump D

D: ----------

jump C

C: ----------

jump B

B: ----------

jump F

F: ----------

jump Done

(iii) [1 for removing jumps, 1 for leaving labels in]

Start: ----------

E: ----------

D: ----------

C: ----------

B: ----------

F: ----------

jump Done

Question 6: Register Allocation [10]

a) What is the purpose of liveness analysis? [2]

To determine which variables need their values kept alive simultaneously

b) What is the purpose of the register allocation algorithm? [2]

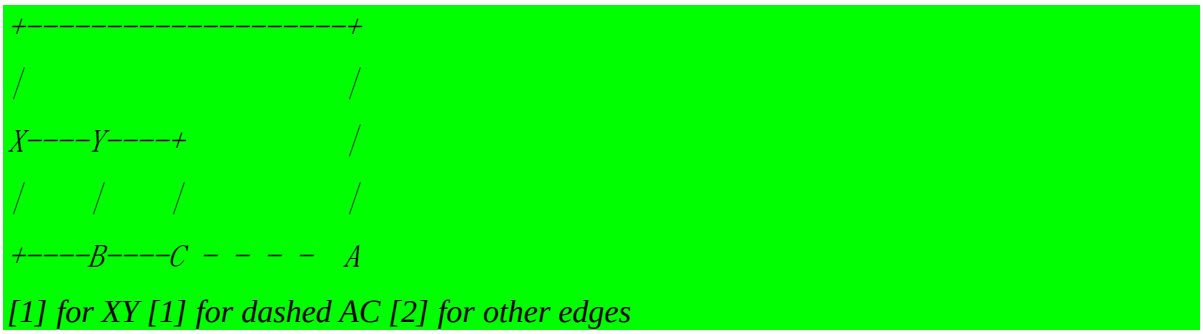
To assign registers to temporaries efficiently

c) Suppose that liveness analysis yields the following live-in/out sets:

```
AX, BX, BCY, BC
```

where X and Y are pre-coloured nodes and A and C are MOVE-related.

Based on just this information, draw a corresponding interference graph. [4]



[1] for XY [1] for dashed AC [2] for other edges

d) Explain how actual spills are dealt with by the register allocation algorithm. [2]

The variable is stored in memory rather than in a register. Each use of the variable is replaced with a new short-lived temporary that first gets the value from memory. Each definition is replaced with a new short-lived temporary that saves the value to memory after it is defined.

Section C Theory of Algorithms [Answer questions 7 and 8 – both compulsory]

Question 7 : You must answer this question. [10]

a) Approximation algorithms are one way of tackling NP-hard problems. Using the discrete knapsack problem as an example, explain the approximation approach. [2]

Use the non-optimal but simple greedy algorithm with an approximation scheme to get required accuracy. e.g. generate all subsets of k items of less. For all those that fill the knapsack, fill in the remainer of the knapsack via greedy approach. subset of highest value is returned.

b) Two very competitive friends have asked you to be a a judge of their programming contest. After a month of work, they reveal their code. Anna has developed a matrix multiplication algorithm of time complexity $O(n^2)$ and Thandi has revealed a graph colouring algorithm of time complexity $O(n^5)$.

i. What are the brute force time complexities for these two algorithms? [2]

n^3 and n!

ii. Should you be impressed with Anna's work? Explain your answer. [2]

Yes - n^2 is the lower bound for matrix multiplication, but there is no known algorithm of this complexity to solve the problem.

iii. Should you be impressed with Thandi's work? Explain your answer.[2]

Yes – n^5 polynomial and the only previous solution were exponential.

iv. Who wins the prize? Explain your decision. [2]

Thandi – she has proved P=NP!

Question 8: You must answer this question. [10]

a) Match **any TWO** of the three problem-solving techniques below with a problem from the list that follows. In each case, define the problem clearly and state briefly how you would apply the associated technique.

i. decrease-and-conquer

ii. transform-and-conquer

iii. dynamic programming

List of problems:

Binary coefficients; Fake coin; GCD (Greatest Common Divisor); Interpolation search; LCM (Lowest Common Multiple); Optimal binary search tree; River-crossing; Transitive closure of directed graphs.

[6]

Any two of:

1 = Fake coin / GCD / Interpolation search (and what is the problem and how to solve it)

2 = LCM / River-crossing (and what and how to solve)

3 = Binary coefficients / Optimal binary search tree / Transitive closure (and what and how to solve)

b) Given a (very large) data file of 2-digit codes in random order, and a query file consisting of code ranges, you must find how many times the codes in each range occur in the data file. Example:

Data file: 61 67 61 67 61 64
Query file:
32 43
61 67
61 64
Output: 0 6 4

(because 32-43 occur 0 times in the data file; 61-67 occur 6 times; and 61-64 occur 4 times).

i. Briefly describe a brute force algorithm for solving this problem.

ii. What is the time complexity of the brute force algorithm if there are M codes in the data file and N ranges in the query file?

iii. Briefly describe a more efficient algorithm for solving this problem.

[4]

For each range in turn go through the entire data file and keep a running total

*M * N*

Go through the data file once, updating counts for each possible value in arr[0..99]; then go through the query file once using arr to calculate answers.

Section D THEORY OF ALGORITHMS **[Answer any 3 questions ONLY]**

Question 9: Greedy algorithms [10]

a) Consider the problem of scheduling *n* jobs of known durations $t_1,t_2,t_3,....,t_n$ for execution by a single processor. The jobs can be excuted in any order, one at a time. You want to find a schedule that minimizes the total time spent by all the jobs in the system. (The time spent by one job in the system is the sum of the time spent by this job in waiting plus the time spent on its execution.)

i. Write down, in pseudocode, a greedy algorithm for this problem. [3]

//Input: list of job times

//output: list of jobs in oder of schedue

sort jobs in non-decreasing order

schedule in order

ii. What is the time complexity of this greedy algorithm? Justify your answer. [2]

nlogn – sorting is primary (only) operation

iii. What would the time complexity be of a brute-force algorithm that performed an exhaustive seach of all possible schedules? [1]

n!

iv. Prove that the greedy algorithm always yields an optimal solution for this scheduling problem. [4]

Proof by contradiction: Assume that there exists an optimal schedule with total time T where jobs are NOT in non-decreasing order. Then, for at least one i, $t_i > t_{(i+1)}$ -> 0> $t_{(i+1)}-t_i$. Swap t_i and $t(i+1)$, then T' =T-t_i + $t_{(i+1)}$. It follows that T' > T. Contradiction.

Question 10 : Lower bounds and backtracking[10]

a) Consider the problem of finding the median of a three-element set {a,b,c}.

i. What is the information-theoretic lower bound for a comparison-based algorithm solving this problem? Show your working. [1]

ceil(log_2(leaves)) = ceil(log_2(3)) = 2 comparisons

ii. Draw a decision tree for an algorithm solving this problem. [3]

a valid decision tree.

b) Prove, with an adversary argument, that any comparison-based algorithm for finding the largest amoung *n* given numbers must make *n-1* comparisons in the worst case. [3]

assume that all elements are distinct -> unique solution. Adversary emp[loys the following rule: reply tru to comparison $a_i > b_j$ iff i>j. Then only correct output will be element n. To produce this, the adversary would have to compare the maximum (element n) to all other n-1 elements (n-1 comparisons). If one of these has not been made, eg (n-1) has not been compared to n, then the anser n-1 is c compatible with the comparisons. Therefore, n-1 is indeed a lower bound.

c) Apply backtracking to solve the following instance of the subset-sum problem: S={1,2,5,6,8} and *d =9.* [3]

have correct graph, with backtracks at {1,2,5} -> soln at {1,2,6}, deadend at {1,2,8}, backtrack to {1} -> {1,5}, backtrack at {1,5,6} etc.

Question 11 : Pattern matching [10]

Consider the problem of **searching for the pattern** ED1C0DED in a hex (hexadecimal) string.

a) How many character comparisons would the brute force algorithm take in searching for ED1C0DED in the string 1212ED1C0DED ?

12

b) Give an example of an 8-character pattern and a 12 character string that would give the worst case performance when brute force is used.

11111112 and 111111111112 (or equiv.)

c) Give the values of elements D, E and F of the bad-symbol shift table as used by the Horspool and Boyer-Moore algorithms when searching for ED1C0DED

D:2; E:1; F:8

d) Show the good-suffix table of the Boyer-Moore algorithm used in searching for ED1C0DED

2, 7, 7, 7, 7, 7, 7

e) How many character comparisons would the Horspool or the Boyer-Moore algorithm take in searching for ED1C0DED in the string 1212ED1C0DED ?

9

f) Describe/characterise the search patterns for which Boyer-Moore and Horspool's algorithms perform exactly the same

when all good-suffix values are identical i.e. when the end of the pattern does not recur or recurs only at the start of the pattern

g) Another common problem besides pattern matching is finding the best path between 2 nodes in a graph.

i. What type of problem-solving technique is used in Breadth First Search?

Decrease-and-conquer

ii. When would you use Breadth First Search and when would you use Floyd's algorithm (i.e. how do you know which to use when)?

BFS=undirected, no weights, just fewest edge paths; Floyd = directed, weighted graph

[10]

Question 12 : Divide-and-Conquer and Dynamic Programming [10]

a) Briefly explain the divide-and-conquer problem solving technique, using any example problem to illustrate your answer.

[2]

1 mark for explanation, 1 mark for example

b) A (huge) data file gives the costs of N items, with the cost of item 1 first, then the cost of item 2, …, and the cost of item N last. A query file consists of several integers Mj. For each query Mj, you must find the minimum cost of buying any Mj consecutive items. Example:

Data file: 4 6 9 11 7

Query file: 3 4
Output: 19 30

(4+6+9 = 19; 6+9+11 = 26; 9+11+7 = 27 so 19 is the minimum of consecutive threesomes;
4+6+9+11=30; 6+9+11+7 = 33 so 30 is the minimum of consecutive foursomes)

i. Briefly describe a brute force algorithm for solving this problem.

ii. Briefly describe a more efficient algorithm for solving this problem.

[4]

1. For each query in turn go through the whole input file, totalling each sequence and finding min.

2. Use 2D array where A[j,k] is cost of sequence j…k ; A[j,k] = A[j,k-1] + A[k,k]; start with diagonal and work towards top right hand corner; once A is filled can find min of M consecutive items by looking for min over A[1,m] … A[n-m-1,n]

c) Show the **array contents** when dynamic programming is used to solve

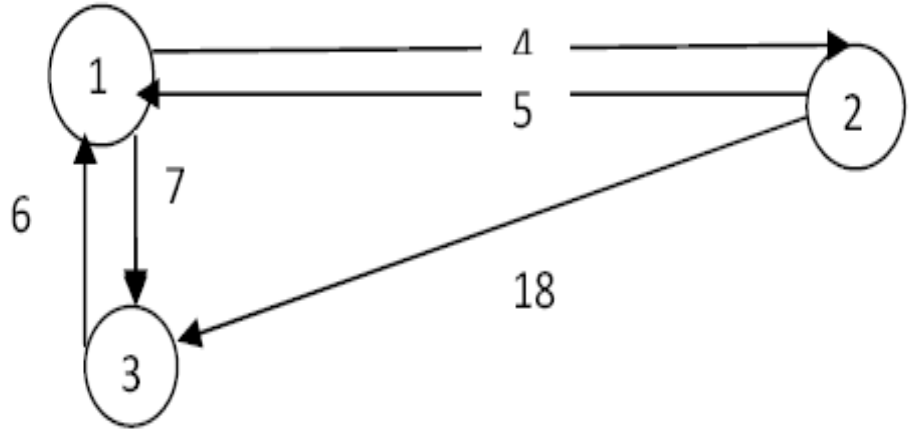
EITHER (1) the **knapsack** problem below
OR (2) the all-pairs shortest path problem below using **Floyd**'s algorithm

i. Item 1 weighs 2kg and is worth R6; item 2 weighs 1kg and is worth R3; item 3 weighs 2kg and is worth R8. The knapsack can hold at most 3kg. What is the greatest Rand value it can hold?

OR

ii. Show **only the first two arrays** in applying Floyd's algorithm to this graph:

0 0 0 0 is knapsack answer; Floyd answer: 0 4 7 and then 0 4 7
0 0 6 6 5 0 18 5 0 12
0 3 6 9 6 99 0 6 10 0
0 3 8 11



[4]