Student ID: .....

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2012

TRIMESTER 2

COMP 261
ALGORITHMS and DATA STRUCTURES

Time Allowed: 3 Hours

Instructions: Attempt ALL Questions.

Answer in the appropriate boxes if possible — if you write your answer elsewhere, make it clear where your answer can be found.

The exam will be marked out of 180 marks.

Only silent non-programmable calculators or silent programmable calculators with their memories cleared are permitted in this examination.

Non-electronic foreign language dictionaries are permitted.

Questions	**Marks**
1. Graphics	[30]
2. Graph Algorithms	[40]
3. Parsing	[35]
4. B+ Trees	[35]
5. String Searching	[15]
6. Finite State Automata and Regular Expressions	[25]

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 1. Graphics [30 marks]

One of the following convolution matrices represents a blur filter, and the other represents a sharpening filter.

-0.04	-0.09	-0.04
-0.09	1.52	-0.09
-0.04	-0.09	-0.04

(A)

0.04	0.09	0.04
0.09	0.48	0.09
0.04	0.09	0.04

(B)

(a) [5 marks] State which matrix is which, and justify your answer.

(b) [5 marks] Explain why it is important that the values in the blur matrix sum to 1.0.

(Question 1 continued on next page)

(Question 1 continued)

(c) [5 marks] Give pseudocode (or Java code) of the algorithm for applying an $m \times m$ convolution matrix to an $n \times n$ image.

(d) [5 marks] The algorithm for rendering 3D models presented in the lectures produced images showing the individual polygons as flat faces with sharp edges. Explain what would be required to render the model as a smoothly rounded surface.

(Question 1 continued on next page)

(Question 1 continued)

(e) [10 marks] Suppose your Z-buffer rendering program has processed some of the polygons in a model, and is now processing a new polygon. The colour of the polygon should be (30,20,240). Part of the edgelists for the polygon are:

	left		right	
y	x	z	x	z
20	12	20	15	17
21	11	21	15	15

The current contents of the Z-buffer are the following. Each cell of the Z-buffer contains a colour and a z value (z increases away from the viewer).

y \ x	10	11	12	13	14	15	16
20	0,0,0 / ∞	0,0,0 / ∞	0,0,0 / ∞	0,0,0 / ∞	120,90,0 / 9	120,90,0 / 10	120,90,0 / 10
21	10,50,100 / 30	10,50,100 / 30	10,50,100 / 31	0,0,0 / ∞	0,0,0 / ∞	0,0,0 / ∞	0,0,0 / ∞

Show the contents of the Z-buffer after your program has rendered the edgelist data into the Z-buffer. State any assumptions you are making.

y \ x	10	11	12	13	14	15	16
20							
21							

Question 2. Graph Algorithms [40 marks]

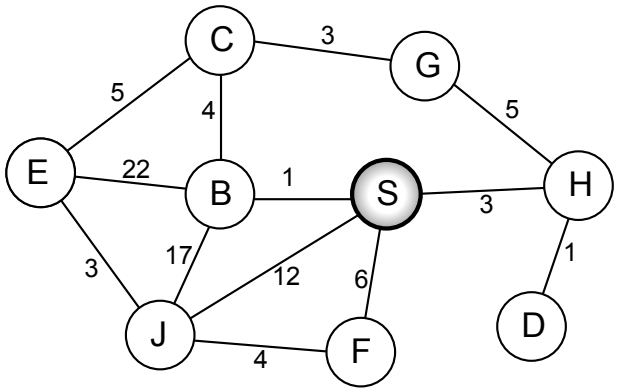
(a) [8 marks] Prim's algorithm (for minimum spanning trees), Dijkstra's algorithm (for shortest paths), and A* search (for shortest path) are all based on the same graph search algorithm. Outline this underlying search algorithm and explain how the three algorithms differ.

(Question 2 continued on next page)

(Question 2 continued)

(b) [17 marks] Show the state of the queue at each step as Dijkstra's algorithm finds **all** the shortest paths from the node S in the following graph.

- At each step, identify the node that is removed from the fringe, and any new nodes that are added.
- Show the fringe entries in the form X/Y(44) to mean node X, added from Y, with priority 44.
- Mark on the graph the path pointers from each node.
- Stop when all nodes are visited.

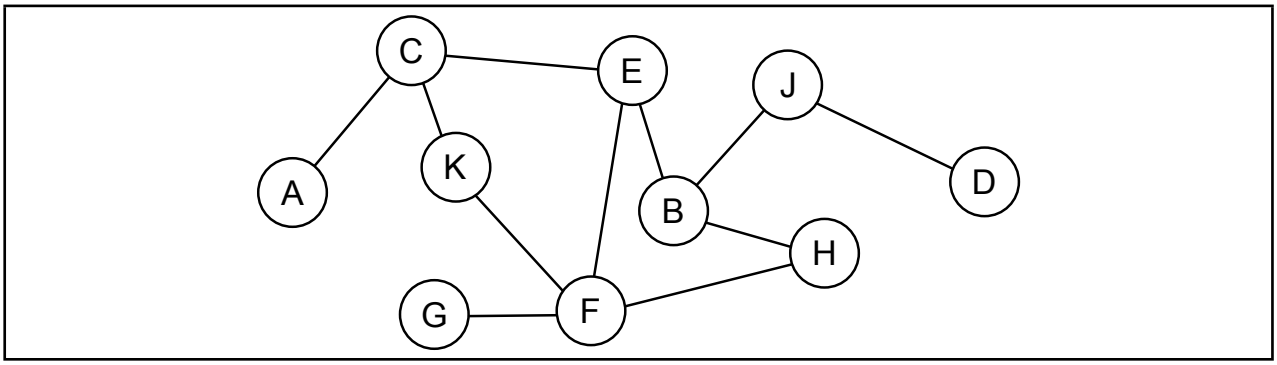


(Question 2 continued on next page)

(Question 2 continued)

(c) [3 marks] Define an articulation point of a graph.

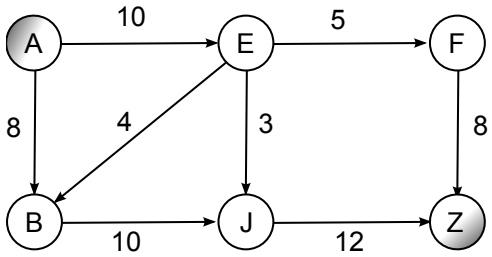
(d) [2 marks] Identify the articulation points of the following graph.

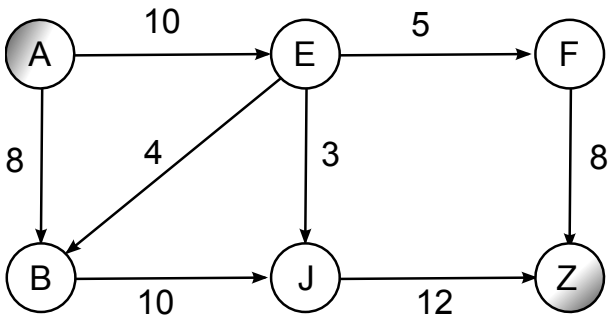


(Question 2 continued on next page)

(Question 2 continued)

(e) [10 marks] Show how the Edmonds-Karp algorithm for Maximum Flow would find the maximum flow through the following graph from the node A to the node Z. Show each path, and how the assigned flow and remaining capacity change through the algorithm.





Maximum Flow =

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Question 3. Parsing [35 marks]

Consider the following grammar for a simple script language

```
SCRIPT ::=  "end"  | COMMAND ";" SCRIPT
COMMAND ::= "run" PROGRAM | "list" FILE
PROGRAM ::= any alphabetic token.
FILE ::= any alphabetic token.
```

(a) [10 marks] Give an abstract syntax tree for the following script, stating what assumptions you made in deciding what to include and what to ignore in the abstract syntax tree.

```
run wget ; list log ; end
```

(Question 3 continued on next page)

(Question 3 continued)

(b) [20 marks] Suppose you are writing a recursive descent parser for the script language (repeated below). You have the following class definitions:

```
public class ScriptNode extends Node{
    private CommandNode command;
    private ScriptNode next;
    public ScriptNode(){ ... }                    // create an empty script node
    public ScriptNode(CommandNode cmd, ScriptNode next){ ... }
}
public class CommandNode extends Node{
    private String action;
    private String argument;
    public CommandNode(String action, String argument){ ... }
}
```

Complete the two methods, parseScriptNode and parseCommandNode on the facing page, that have a Scanner parameter. Each method should return an appropriate Node of a parse tree, if the Scanner contains a valid script or command, and should return null otherwise.

Assume that the Scanner has an appropriate delimiter for separating the tokens. The methods do not need to generate any error messages.

Your methods may (if you wish) use the gobble method defined below.
Note also that s.hasNext("[a-zA-Z]+") will be true iff the next token in s is an alphabetic token.

```
private boolean gobble(String pat, Scanner s){
    if (s.hasNext(pat)) {s.next(); return true;}
    return false;
}
```

The grammar (repeated):

```
SCRIPT ::=  "end"  | COMMAND ";" SCRIPT
COMMAND ::= "run" PROGRAM | "list" FILE
PROGRAM ::= any alphabetic token.
FILE ::= any alphabetic token.
```

(Question 3 continued on next page)

(Question 3 continued)

```
public ScriptNode parseScriptNode(Scanner s){




}
public CommandNode parseCommandNode(Scanner s){




}
```

(Question 3 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 3 continued)

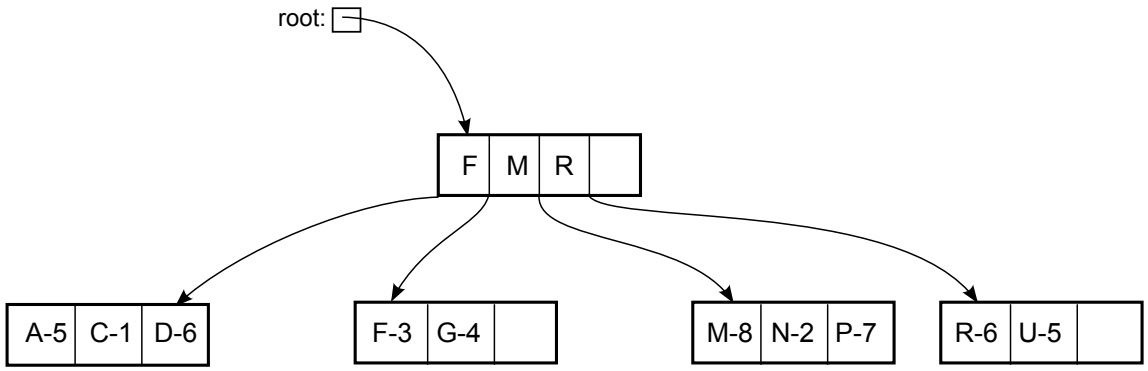
(c) [5 marks] Suppose we wanted to add another option to the Command rule:

```
COMMAND ::= "run" PROGRAM | "list" FILE | COMMAND ">" FILE
```

Explain why it would now be problematic to turn the script grammar directly into a recursive descent parser.

Question 4. B+ Trees [35 marks]

(a) [15 marks] The following B+ tree has internal nodes that hold up to 4 keys, and leaf nodes that hold 3 key-value pairs. (The keys are letters; the values are numbers)



Show how the tree is modified as the following key-value pairs are added:

```
J-5  S-7 E-8 B-9  Q-4
```

(Question 4 continued on next page)

(Question 4 continued)

(b) [5 marks] Explain how the B+ tree algorithms guarantee that a B+ tree will be always remain balanced.

(c) [5 marks] Explain why it is important that the B+ tree algorithm ensures that all internal nodes (except the root node) are kept at least half full.

(Question 4 continued on next page)

(Question 4 continued)

(d) [10 marks] Suppose you are implementing a B+ tree in a file, storing each node of the tree in one block. The keys of the B+ tree are student ID numbers and the values are student names. Each ID number requires 4 bytes, and the student names are at most 26 characters (26 bytes). The blocks are 512 bytes long, and the first byte of each block stores the type of the block.

(i) How many key-value pairs can be stored in a leaf node? Explain your reasoning, show your working, and state any assumptions you make about the information stored in the blocks.

(ii) How many keys can be stored in an internal node? Explain your reasoning, show your working, and state any assumptions you make about the information stored in the blocks.

Question 5. String Searching [15 marks]

(a) [5 marks] The Knuth Morris Pratt algorithm searches for a string in a piece of text. It first analyses the string and builds a table. Explain how the table enables KMP to search more efficiently than the naive string search algorithm.

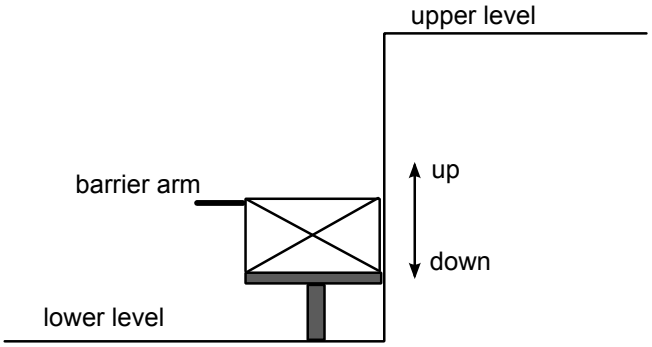
(b) [10 marks] Show the table that the following KMP table building algorithm would construct given the string `"inviting"`:

```
computeKMPTable(string)
    initialise table to an array of integers
    table[0] ← −1      table[1] ← 0;
    pos ← 2;           j ← 0;
    while pos < string.length
        if string[pos−1] = string[j]
            table[pos] ← j+1
            pos++
            j++
        else if j > 0
            j ← table[ j ]
        else
            table[pos] ← 0
            pos++
    return table
```

Question 6. Finite State Automata and Regular Expressions [25 marks]

(a) [15 marks] Complete the design on the facing page of a Finite State Automata controller for a wheelchair lift that goes between two levels.

The lift has a button to request the lift to move, and a safety barrier arm that can be manually opened and closed. If the button is pressed when the lift is at one of the levels and the barrier is closed, then the barrier will be locked and the lift will move to the other level. Once it arrives at the other level, the barrier will be unlocked.



The partial design shows the six possible states of the FSA controller, and two of the possible transitions. You are to add the remaining transitions, and label each transition with the sensor values that would cause the transition to occur and any action or actions that should be performed.
For example, a transition might be labeled

"at lower $\rightarrow$ up, unlock"

to mean that if the lower level sensor becomes true, the controller will change state and perform the up action and the unlock action. (Not actually a good idea!)

There are four actions that the controller can perform on the lift:

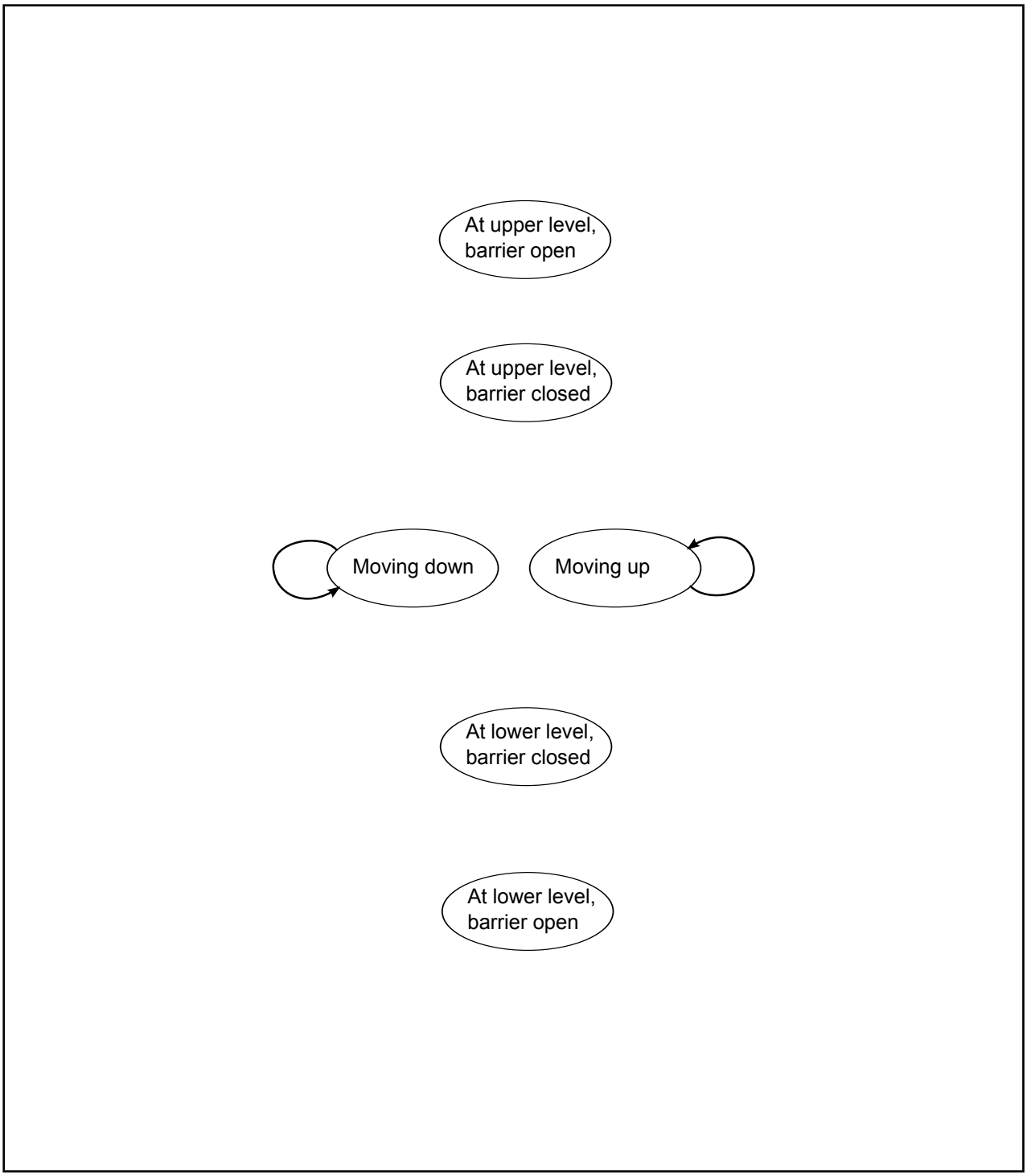
- lock the barrier arm (only possible when the barrier is closed).
- unlock the barrier arm (should not be done when the lift is between levels).
- up, which causes the lift to move upwards (unless it is already at the upper level).
- down, which causes the lift to move downwards (unless it is already at the lower level).

The lift has four input buttons/sensors:

- closed: true iff the barrier arm has been closed, false otherwise.
- at lower: true iff the lift is at the lower level
- at upper: true iff the lift is at the upper level
- request: true iff the user has pressed the move button.

(Question 6 continued on next page)

(Question 6 continued)



(Question 6 continued on next page)

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

(Question 6 continued)

(b) [10 marks] Write a regular expression for matching phone numbers. A phone number has an optional area code (0 followed by one or more digits, and possibly surrounded by round brackets), an exchange code (three digits), and number (three or four digits). The first two components are separated by a space, and the last two components by a hyphen. For example, your expression should match the following phone numbers:

```
(04) 123-5894
(023) 456-789
05 321-7654
456-7654
```

Hint: See the documentation on regular expressions at the back of the exam.

****

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

1 Regular Expression Documentation

Assume X and Y are patterns, n and m are integers

`[abc0-9]`	matches any one of the listed characters
`\(`	matches a open round bracket
`\)`	matches a closing round bracket
`\s`	matches a whitespace character
`\d`	matches a digit: `[0-9]`
`\w`	matches an alphanumeric character: `[a-zA-Z_0-9]\verb`
`^`	matches the beginning of a line
`$`	matches the end of a line
`\b`	matches a word boundary (between a word and a non-word character)
X`?`	matches X once or not at all
$X\star$	matches X zero or more times
X`+`	matches X one or more times
X{n,m}	matches X at least n but not more than m times
XY	matches X followed by Y
X`|`Y	matches either X or Y
`(`X`)`	matches X as a group
`\1`, `\2` etc	matches the same thing as the 1st (2nd, ...) group.