PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Final Examination: Semester 1 **Academic Year**: 2010-2011

Date: 7th October, 2010 **Time**: 9:00 – 12:00 (**3 hours**)

Subject Number: 241-303 **and** 240-304 **Rooms**: Robot Head *and* A401

Subject Title: Discrete Mathematics
and: Mathematics for Computer Engineering

Lecturer: Aj. Andrew Davison

Exam Duration: 3 hours

This paper has 5 pages.

Authorized Materials:

- Writing instruments (e.g. pens, pencils).
- Books (e.g. dictionaries) and calculators are **not** permitted.

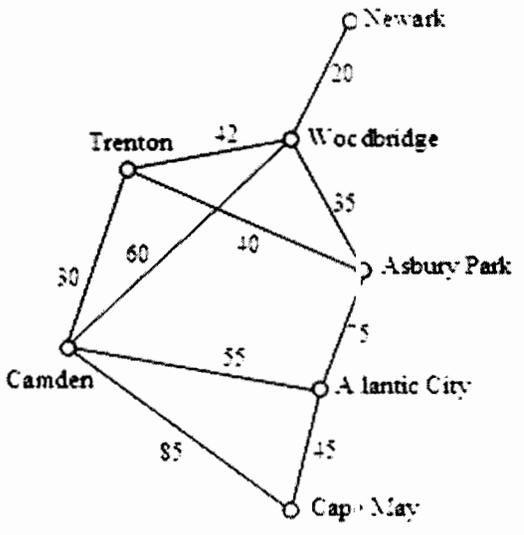
Instructions to Students:

- *Answer questions in English.* Perfect English is **not** required.
- Attempt all questions.
- Write your answers in an answer book.
- Start your answer to each question on a new page.
- Clearly number your answers.
- Any unreadable parts will be considered wrong.
- When writing programs, use good layout, and short comments; marks will not be deducted for minor syntax errors.
- The marks for each part of a question are given in brackets (...).

Question 1

(20 marks; 20 minutes)

Consider the following weighted graph of distances between major cities in New Jersey, USA.

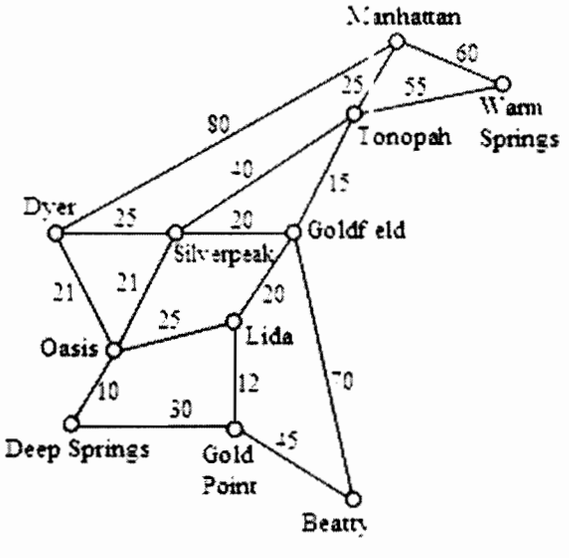


Using Dijkstra's algorithm, find the shortest distances from **Newark** to **Cape May**. Show **all** your working. Clearly indicate the path that makes up the shortest distance.

Question 2

(20 marks; 20 minutes)

The following graph shows dirt roads that connect small towns. The lengths of the roads are shown as edge weights.



a) Which roads should be covered in concrete so there are good roads between all the towns, and the smallest amount of concrete is used? (10)

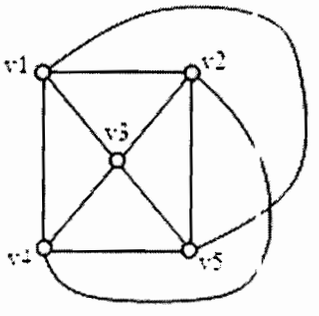
Show all your working. *Hint*: you do **NOT** need to cover all the roads in concrete.

b) There are two ways of solving part (a). Explain the other approach, and calculate whether it gives the same or a different answer. (10)

Question 3

(30 marks; 30 minutes)

a) Draw an adjacency list for the following graph. (5)

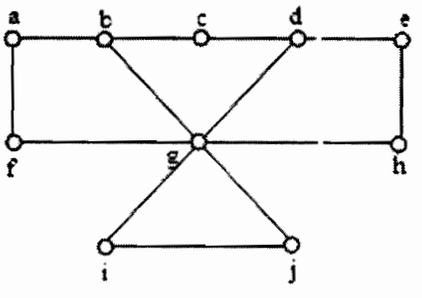


b) Give the ANSI C data types for the adjacency list in part (a). (5)

c) Explain in words the Euler cycle **and** Hamiltonian cycle. (10)

d) Explain whether the graph in part (a) has an Euler cycle. (5)

e) Show that the following graph does not contain a Hamiltonian cycle. (5)

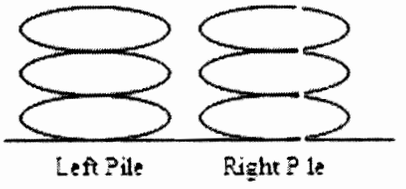


Question 4 is on the next page.

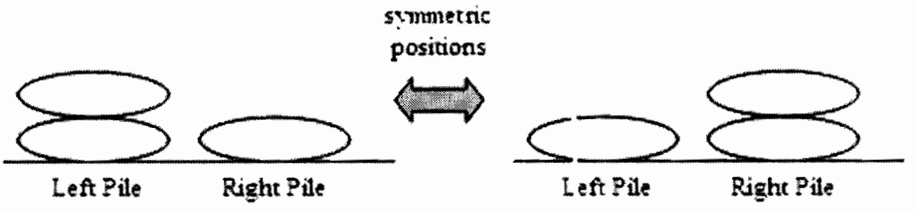
Question 4

(60 marks; 60 minutes)

a) Draw a game tree for Nim in which the initial position consists of two piles of three stones each, as shown below. Each player can take one or more stones from one pile, and the player who removes the last stone loses. (20)



You should reduce the size of the game tree by ignoring extra *symmetric positions*. An example of two symmetric positions is shown below:

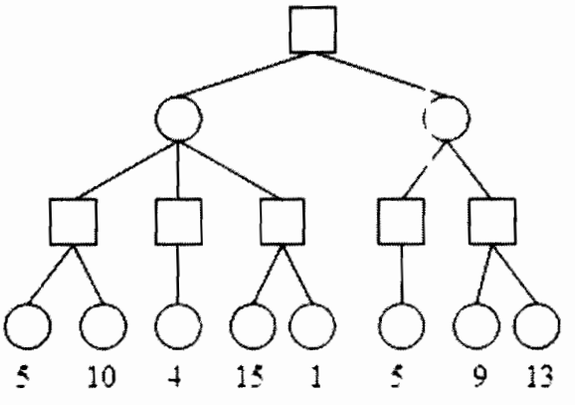


'Symmetric' means that the piles in one position are the same as in the other position after the left and right sides have been switched. Only one of these positions needs to be included in the game tree; the other one can be left out of the tree.

b) Assign scores to the game tree positions in part (a) to indicate whether the first player will win or lose. Explain all your working. (20)

c) Explain alpha-beta pruning in words, with the aid of small diagrams. (10)

d) Apply alpha-beta pruning to the following graph, clearly indicating which subtrees (if any) are pruned. (10)



Question 5

(20 marks; 20 minutes)

a) Draw a finite state automaton which will accept input containing any number of 'a's but **only one 'b'**. (5)

b) Draw a finite state automaton which will accept input containing any number of 'a's and **at least one 'b'**. (5)

c) Draw a finite state automaton which will accept input containing **at least two 'a's** and any number of 'b's. (10)

Question 6

(30 marks; 30 minutes)

Consider the following grammar for a word made up of lowercase letters:

Word → Letter Word | ε
Letter → a | b | c | ... | x | y | z

a) Translate the grammar into syntax graphs. (5)

b) Translate the syntax graphs into a parser. The parser should print "yes" if the input string matches the grammar; "no" otherwise. The parser should **not** build a parse tree. (15)

c) What problems will occur with translating the following grammar rule into a parser function: (5)

WordProb → WordProb Letter | Letter

d) Explain how to translate the grammar rule in (c) into a more suitable form for parsing. Do **not** write syntax graphs or a parser, only the modified grammar. (5)

--- End of Examination ---