SCHOOL OF SCIENCE

GMIT
GALWAY-MAYO INSTITUTE OF TECHNOLOGY
INSTITIUID TEICNEOLAÍOCHTA NA GAILLIMHE-MAIGH EO

GMIT EXAMINATIONS
SESSION: SUMMER 2014/2015

COURSE: B. Sc. (Hons) in SOFTWARE DEVELOPMENT **YEAR/STAGE:** AWARD

SUBJECT: THEORY OF ALGORITHMS

Date: Fri. 8th May
Time: 2.30 p.m.

EXTERNAL EXAMINERS: MR. TOM DAVIS
DR. MICHAEL SCHUKAT

INTERNAL EXAMINERS: MR. M. FITZGERALD

TIME ALLOWED: 2 HOURS

INSTRUCTIONS TO CANDIDATES:

Answer THREE QUESTIONS
All questions carry equal marks

SPECIAL REQUIREMENTS:

1. (a) Using truth tables or otherwise show that

(i) $P \vee (P \wedge Q) \equiv P$ (ii) $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$

Prove the following equivalences using Boolean algebra:

(iii) $(A \Rightarrow B) \wedge (A \Rightarrow \neg B) \equiv \neg A$

(iv) $A \Rightarrow (B \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$

(v) $(A \Rightarrow B) \wedge (\neg A \Rightarrow B) \equiv B$ (9 marks)

(b) If L is a set of lecturers and S a set of students and for $p \in L$ and $s \in S$ the predicates lec(p,s)≡"p lectures student s", and compsc(p) ≡ "p is a computer science lecturer" then express formally the following propositions:

(i) There is a student who is lectured by all the lecturers

(ii) There is a student who has no computer science lecturer

(iii) There is a computer science lecturer who has no students

(iv) Some students have all the computer science lecturers (8 marks)

(c) The following is a set of preconditions for an operation where P_1, P_2, P_3 are independent predicates: $\{P_1 \wedge P_2, \neg P_1 \wedge P_2, P_1 \wedge \neg P_2 \wedge P_3, \neg P_1 \wedge \neg P_2 \wedge P_3\}$

Draw a precondition decomposition diagram to show that they are mutually exclusive and hence identify the missing precondition to ensure exhaustivity. (8 marks)

2. We require here to specify a motion control program/module for an elevator. The elevator moves between floors 1(ground) and m(top) floors and at all times the current elevator position is p($1 \leq p \leq m$). Floor requests are received and stored in request buffers C(elevator car), U(up from a floor), D(down from a floor) independently filled but available to and deletable by the module. When approaching a floor f from below at a predetermined checkpoint in advance by e($0<e<1$), with position f–e, of the floor being reached the module makes a decision to either stop at that floor f by setting the floor destination variable p' to f or to skip that floor to the next upward advance position p'=f+1–e and analagously when approaching a floor f from above. When stopped at a floor f the module can choose to subsequently move upward by setting p' to f+1–e, downward by setting p' to f–1+e or remain stationary by setting p' to f. Further the elevator operates in either upward mode(d=1), or downward mode(d=-1) which is set by the module. If in upward mode it continues upward in that mode until all requests from above are dealt with and similarly if in downward mode. Further, for greater efficiency, if in upward mode it will not stop at a floor from which there is only a downward request if there are requests from other floors above. In that situation it will visit that floor and deal with that downward request when it has reversed direction and heading downwards and analagously when in downward mode. The following are the schemas for the upward mode d=1 when stopped at floor p.

PTO

Q2 Continued……..

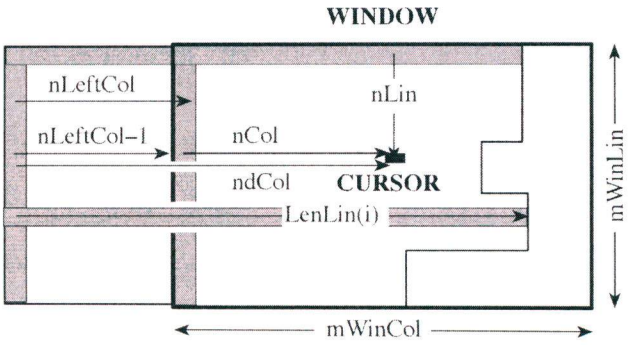
LiftUpMoveUp	
d=1	*going upward*
p∈F	*at floor p*
∃q∈R[q>p]	*there is a request from floor above p*
p' = p+1–e	*move to decision stage before next floor up*
(C',U',D') = (C,U,D)/{p}	*remove p from all request sets*
d' = d	*still upward*

LiftUpReverse	
d=1	*going upward*
p∈F	*at floor p*
¬∃q∈R[q>p]	*there is no request for floor above p*
∃q∈R[q<p]	*there is a request for floor below p*
p' = p–1+e	*move to decision stage before next floor down*
(C',U',D') = (C,U,D)/{p}	*remove p from all request sets*
d' = -d	*reverse direction*

LiftUpIdle	
d=1	*going upward*
p∈F	*at floor p*
¬∃q∈R[q>p]	*there is no request for floor above p*
¬∃q∈R[q<p]	*there is no request for floor below p*
p' = p	*stay put*
(C',U',D') = (C,U,D)/{p}	*remove p from all request sets*
d' = d	*same direction*

Complete a formal specification for the upward mode d=1 when at a checkpoint p before floor p+e and draw relevant portion i.e. d=1 of the precondition decomposition diagram in support. (25 marks)

3. (a) We wish to formally specify the cursor control operation of a basic text editor with fixed width typeface. The view of the document is from a window with a fixed number of text lines(mWinLin) and columns(mWinCol). The current document is composed of nDocLin lines of text of variable length expressed through the function LenLin(i) = length(number of characters in) line i. The position of the window on the document is determined by the document number of the line at the top of the screen nTopLin(≥1) and of the column nLeftCol(≥1) at the left of the screen. There must be at least one line of text on the screen at all times. The cursor position is determined by the screen line(nLin) and column(nCol) numbers and must be on the text and on the screen at all times . The document is left justified (ragged right) as in the diagram.



PTO

Q3 Continued.....

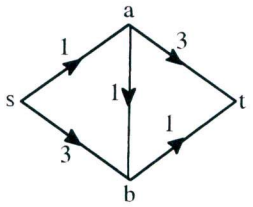
Up Key: The Up key moves up one ine and in the same column, if possible, otherwise moving in to end of the line above with horizontal/vertical scrolls only as required. If at the top of the document it does not move.

Formally specify its action **when at the top of the screen i.e. nLin=1**.

(b) For the following sample scenarios calculate the variable changes caused by the Up key:
nLin=1, LenLin(14)=18, LenLin(15)=25, LenLin(16)=25, LenLin(17)=25, LenLin(18)=35
(i) nLeftCol=9, nTopLin=15, nCol=12 (ii) nLeftCol=11, nTopLin=17, nCol=13
(iii) nLeftCol=30, nTopLin=18, nCol=5 (25 marks)

4. (a) State the basic requirements for a valid recursive function definition. (4 marks)

(b) Given a weighted directed graph G, with nodes N(G) and edges E(G), and distance(weight) function d:E(G)→R define a recursive function to find the length of the shortest path between two nodes **s**,**t**. Does it satisfy condition (a) and if so why? Apply it to the find the length of the shortest path from **s** to **t** in the graph across:



(15 marks)

(c) The following is a recursive boolean valued function with a pair of lists for arguments:

$$
\begin{aligned}
F(L_1, L_2) &= F(tl(L_1), tl(L_2)) && ,\#L_1 \geq 1 \wedge \#L_2 \geq 1 \wedge (hd(L_1) = hd(L_2) \vee hd(L_1) = *) \\
&= F && ,\#L_1 \geq 1 \wedge \#L_2 \geq 1 \wedge (hd(L_1) \neq hd(L_2) \wedge hd(L_1) \neq *) \\
&= F && ,\#L_1 \geq 1 \wedge \#L_2 = 0 \\
&= T && ,\#L_1 = 0
\end{aligned}
$$

(i) Does this meet the requirement of definition 4(a)? Justify your answer.
(ii) Evaluate F(uwrx , uwrxyt), F(uwrxyt , uwrx), F(uw*x,uwrxt)
(iii) What does $F(L_1, L_2) = T$ say about L_1, L_2 ? (6 marks)

5. (a) (i) Derive an expression for the complexity(no of moves) of the recursive algorithm to solve the general Towers of Hanoi problem.
(ii) How long will it take a computer to enumerate the transformation for n=60 assuming it records the transactions at a rate of 100,000 per second. (10^7seconds = 1 year).
(10 marks)

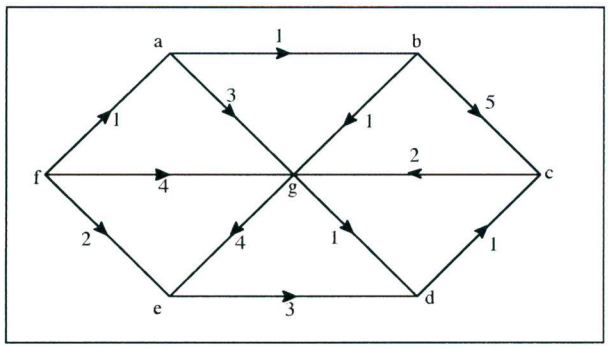
(b) Prove that that there is no shorter algorithm. (15 marks)

PTO

Continued……..

6. (a) State and prove Dijkstra's algorithm for finding the shortest path between two nodes in a positively weighted digraph. (15 marks)

(b) Apply it to the graph below to find the shortest distance and route from node **f** to node **c**



(10 marks)