A20483 *No Calculator permitted in this examination*

UNIVERSITY OF BIRMINGHAM

School of Computer Science

First Year – Degree of BSc with Honours

Artificial Intelligence and Computer Science

Computer Science

Computer Science with Study Abroad

Computer Science with Business Management

Natural Sciences

First Year – Degree of BEng/MEng with Honours

Computer Science/Software Engineering

First Year – Joint Degree of BSc/MSci with Honours

Mathematics and Computer Science

06 18187

Foundations of Computer Science

Main Examinations 2008

Time allowed: 3 hours

[Answer ALL Questions]

[PART A. USE SEPARATE ANSWER BOOK.]

Question 1 [SHORT QUESTIONS] [17% in total]

(a) What is 372_8 in decimal? [2%]

(b) Put 11000101110_2 into hexadecimal. [2%]

(c) Multiply $11010_2 \times 10111_2$ in binary. [3%]

(d) Multiply 3.8×10^{17} by 1.3×10^{-29}, giving your answer with mantissa to one decimal place. [2%]

(e) A CPU is connected to a memory by a 16 line data bus and a 24 line address bus. What is the size of the memory? [2%]

(f) Write $(3 + 7) \times (8 - 2) - 4$ in postfix (reverse Polish) notation. [2%]

(g) What can you say about the time taken by binary search of a sorted array? [2%]

(h) What name is given to the register storing the address of the current instruction? [2%]

Question 2 [QUEUES] [12% in total]

(a) A queue of integers is stored in a circular fashion inside an integer array `a` of length `n`. The variable `front` is the current front index, and the variable `size` is the current length of the queue. Write methods for enqueing and dequeueing a value. [4%]

(b) A department has two printers called `degas` and `renoir`. Each prints at a rate of one page per second.

At time 15:05:00 both printers are idle and have no jobs pending. Between 15:05:00 and 15:10:00 the files sent are shown as follows. Each file is sent simultaneously to both printers.

Time sent	Sender	Category	Filename	No. of pages
15:07:22	ug35trt	student	assignment.pdf	20
15:07:30	ug52mxa	student	jokes.pdf	15
15:08:02	ug30rrr	student	diss.pdf	70
15:08:28	ug19zyr	student	cv.pdf	5
15:08:45	pbl	staff	report.pdf	20
15:09:48	ug53uip	student	news.pdf	17
15:09:55	ug74aec	student	homework.pdf	34

The printer `degas` holds its jobs in a queue.

At what time does each file become available for collection from `degas`? [4%]

(c) The printer `renoir` holds its jobs in two queues: one for staff files, one for student files. Staff files are always given priority. As far as possible, say at what time each file becomes available for collection from `renoir`. [4%]

Question 3 [LINKED LISTS] [12% in total]

(a) State one advantage of storing a list as a linked list rather than an array. [3%]

(b) Here is a piece of memory, containing a linked list of byte-sized integers whose head pointer is at address 30F2.

location	30F2	30F3	30F4	30F5	30F6	30F7
contents	30	FB	29	F2	FF	FF

location	30F8	30F9	30FA	30FB	30FC	30FD
contents	80	30	F5	64	30	FE

location	30FE	30FF	3100	3101	3102	3103
contents	3D	30	F8	08	CC	72

By convention, a null pointer is FFFF.

What are the entries in the list? (You can write each entry as two hexadecimal digits.) [3%]

(c) To insert an entry E4 before the exclusive-3rd entry, how would you change the memory contents? The locations 3101–3103 inclusive are available for you to use. [3%]

(d) Another linked list of byte-sized integers is stored in memory, with head pointer at 722A. The list is not empty. Write assembly code to add 3 (ignoring overflow) to the exclusive-zeroth entry in the list. (An assembly reference guide is provided at the end of this exam paper.) [3%]

Question 4 [ARRAYS] [9% in total]

(a) What check does Java make every time an array entry is read or assigned to? [3%]

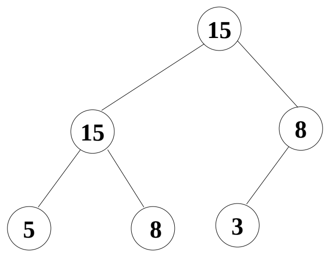
(b) Write a function `sorted` that, when applied to an integer array, returns true if the list of entries is sorted in increasing order, and false otherwise. For example, your function applied to an array storing [3,4,6,6,6,8,8,23] should return true, but applied to an array storing [7,2,4,4,2] it should return false. [6%]

[PART B. USE NEW ANSWER BOOK.]

Question 5 [TREES] [17% in total]

(a) Insert the numbers 1, 2, 3, 4, 5, 6, 7 in this order in an initially empty priority queue which is represented as a heap tree. Display each of the resulting heap trees [5%]

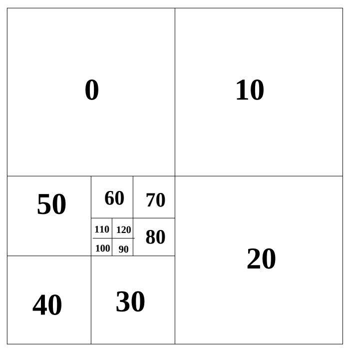
(b) Is the following tree a heap tree? Is it a binary search tree? Explain your answer.



[4%]

(c) Let a quadtree be defined either as a number (in the range 0 to 255) or as a node consisting of the four quadtrees, `lu`, `ll`, `ru`, and `rl`. Assume constructors, `consQT(lu,ll,ru,rl)` and `baseQT(value)`, selectors, `lu(qt)`, `ll(qt)`, `ru(qt)`, and `rl(qt)`, as well as a condition, `IsNumber(qt)`, to represent quadtrees.Quadtrees of this type can be used to store grey-value pictures (with 0 representing black and 255 white).

An example is given by



Write an algorithm – making use exclusively of these interface functions – that determines the average brightness of a given picture as a rounded grey-value. [4%]

(d) Are there binary trees which are heap trees and binary search trees at the same time? If no, explain why. If yes, give examples (if possible all examples) and explain. [4%]

Question 6 [SORTING, HASH TABLES] [16% in total]

(a) Let the array $[6, 4, 8, 3, 7, 4]$ be given. Sort the array using `bubblesort`. Show each intermediate array. [4%]

(b) Let the array `a` be given as $[6, 4, 8, 3, 7, 4]$. Sort it using `quicksort`. The pivot should be chosen as the middle element `a`[(`size` − `1`)/2]. Show all important steps such as partitioning and the recursive calls of the algorithm. [4%]

(c) Assume a hashtable represented as an array of size 7, we want to store strings consisting of three digits. The primary hash key is the numerical value of the second digit modulo 7. Insert the following strings into the initially empty hashtable by resolving collision by linear probing: `"007"`, `"137"`, `"737"`, `"555"`, `"371"`, and `"234"`. [4%]

(d) What is the worst case time complexity of `heapsort`? Explain in detail why. [4%]

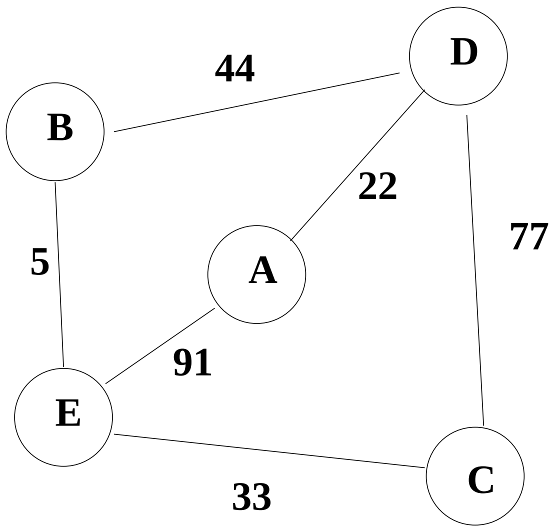
Question 7 [GRAPHS] [17% in total]

(a) Let the following adjacency matrix be given

	A	*B*	*C*	*D*	*E*
A	0	1	1	1	1
B	0	0	0	0	1
C	1	0	0	1	1
D	0	1	1	0	0
E	0	0	1	0	0

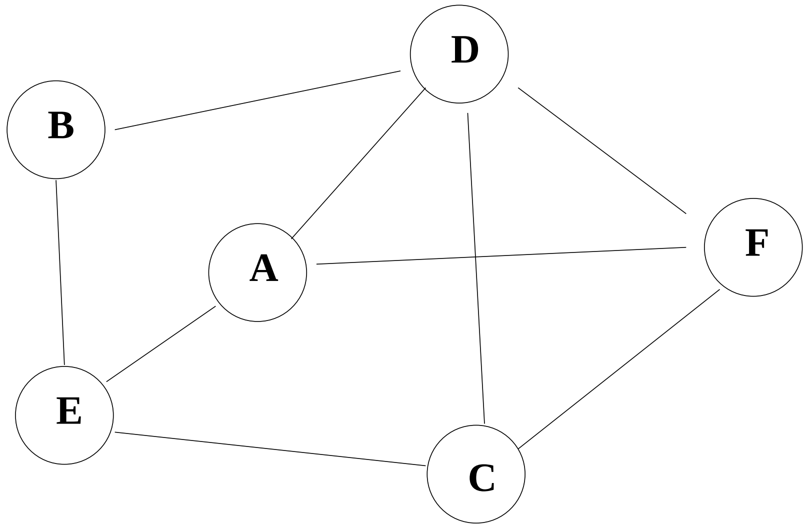
Display the corresponding directed graph. [4%]

(b) Let the following graph be given:



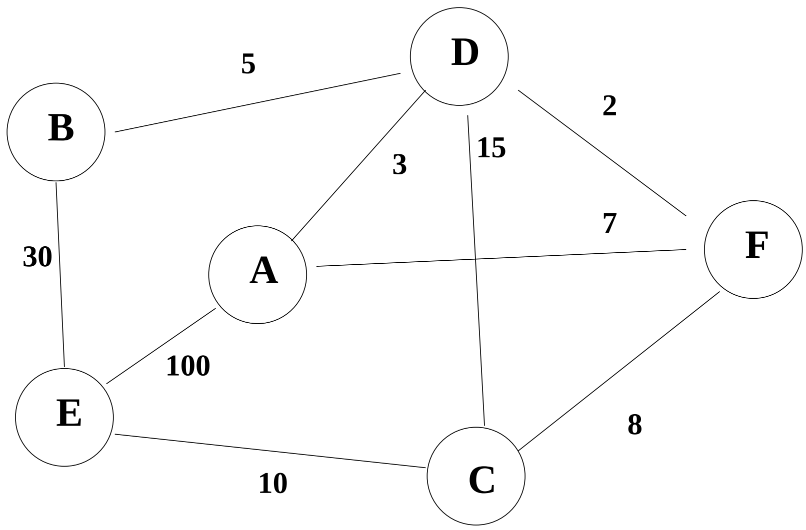
What is the corresponding weight matrix? [4%]

(c) Let the following graph be given:



Give a spanning tree. [4%]

(d) Let the following weighted graph be given:



Use Dijkstra's algorithm to find the shortest path from **A** to **E**. Show all relevant steps in detail. [5%]

ASSEMBLY REFERENCE

The address bus is 16 lines, the data bus is 8 lines, and each instruction occupies 3 bytes.

8-bit registers R0,R1,R2,R3,R4,R5,R6,R7

16-bit registers L0,L1,L2,L3,L4,L5,L6,L7, program counter (PC)

In the table, *n* represents 8 bits; *nn* 16 bits; and *i*, *j*, *k* all represent 0,1,2,3,4,5,6 or 7.

LA R*i* <-- *n*	put *n* into register R*i*, add 3 to PC
LA L*i* <-- *nn*	put *nn* into register L*i*, add 3 to PC
LR R*i* <-- R*j*	copy register R*j* into register R*i*, add 3 to PC
LR L*i* <-- L*j*	copy register L*j* into register L*i*, add 3 to PC
LM R*i* <-- *nn*	load the contents of address *nn* into R*i*, add 3 to PC
LM L*i* <-- *nn*	load the contents of address *nn* and *nn* + 1 into L*i*, add 3 to PC
STM R*i* --> *nn*	store the contents of R*i* at address *nn*, add 3 to PC
STM L*i* --> *nn*	store the contents of L*i* at address *nn* and *nn* + 1, add 3 to PC
LI R*i* <-- R*j*, *nn*	load the contents of address *nn*+R*j* into R*i*, add 3 to PC
STI R*i* --> R*j*, *nn*	store the contents of R*i* at address *nn*+R*j*, add 3 to PC
LRI R*i* <-- L*j*	load the contents of address L*j* into R*i*, add 3 to PC
LRI L*i* <-- L*j*	load the contents of address L*j* and (L*j*) + 1 into L*i*, add 3 to PC
STRI R*i* --> L*j*	store the contents of R*i* at address L*j*, add 3 to PC
STRI L*i* --> L*j*	store the contents of L*i* at address L*j* and (L*j*) + 1, add 3 to PC
COMP R*i*, R*j* --> R*k*	compare R*i* and R*j* as signed integers in 2's complement representation if R*i* >R*j*, put 1 in R*k* , add 3 to PC if R*i* =R*j*, put 0 in R*k*, add 3 to PC if R*i* <R*j*, put -1 in R*k*, add 3 to PC
COMP L*i*, L*j* --> R*k*	compare L*i* and L*j* as natural numbers if L*i* >L*j*, put 1 in R*k*, add 3 to PC if L*i* =L*j*, put 0 in R*k*, add 3 to PC if L*i* <L*j*, put -1 in R*k*, add 3 to PC
JP *nn*	put *nn* into PC
JPR L*i*	copy L*i* into PC
JPG R*i*, *nn*	if R*i* = 1, put *nn* into PC, else add 3 to PC
JPE R*i*, *nn*	if R*i* = 0, put *nn* into PC, else add 3 to PC
JPL R*i*, *nn*	if R*i* = -1, put *nn* into PC, else add 3 to PC
JPGE R*i*, *nn*	if R*i* = 1 or 0, put *nn* into PC, else add 3 to PC
JPLE R*i*, *nn*	if R*i* = -1 or 0, put *nn* into PC, else add 3 to PC
JPNE R*i*, *nn*	if R*i* = -1 or 1, put *nn* into PC, else add 3 to PC
INC R*i*	increment R*i* (ignoring overflow), add 3 to PC
INC L*i*	increment L*i* (ignoring overflow), add 3 to PC
DEC R*i*	decrement R*i* (ignoring overflow), add 3 to PC
DEC L*i*	decrement L*i* (ignoring overflow), add 3 to PC
ADD R*i*, R*j* --> R*k*	put R*i* + R*j* (ignoring overflow) into R*k*, add 3 to PC
SUB R*i*, R*j* --> R*k*	put R*i* - R*j* (ignoring overflow) into R*k*, add 3 to PC
MULT R*i*, R*j* --> R*k*	put R*i* × R*j* (ignoring overflow) into R*k*, add 3 to PC
PUSHA *n*	push *n* (one byte), add 3 to PC
PUSHAL *nn*	push *nn* (high byte, then low byte), add 3 to PC
PUSH R*i*	push contents of R*i* (one byte), add 3 to PC
PUSH L*i*	push contents of L*i* (high byte then low byte), add 3 to PC
POP R*i*	pop top byte of stack into R*i*, add 3 to PC
POP L*i*	pop top byte of stack into low byte of L*i*, then pop next byte of stack into high byte of L*i*, add 3 to PC