Family Name:	
Other Names:	
Signature:	
Student Number:	

This PAPER is NOT to be retained by the STUDENT

The University Of New South Wales

COMP1917 Final Exam

Higher Computing hs1917

July 2009

Time allowed: **3 hrs**
Total number of questions: **21**
Total number of marks: **104**

You must hand in this entire exam paper and ALL the answer booklets. Otherwise you will get zero marks for the exam and a possible charge of academic misconduct.

Ensure that you fill in all of the details on the front of this pink paper and each of the answer booklets, and then SIGN everything. Mark one booklet PART B, one PART C, one PART D, and one WORKING ONLY. The WORKING ONLY booklet is not marked but you must still return it at the end of the exam.

Do not use red pen or pencil in your answers. You can only use red pen or pencil in your WORKING ONLY booklet. Unless advised otherwise you may not use any language features or library functions not covered in class. Code which does not comply with the course style guide, which contains a security vulnerability, or which is overly complex or unclear will be penalised.

There are two marks for following the examination instructions.
Questions (and sub-questions) are not worth equal marks.
You will be provided with a printed set of reference notes which you may keep. Calculators may not be used.
Answer all questions. Only answer one sub-question in the final question of part D.

Examiner's Use Only:

Followed Instructions?	Part A	Part B	Part C	Part D	Potato?	**Total**

Part A: Short Answer Questions

Answer these questions in the spaces provided on **this pink question paper**. DO NOT answer these questions in an **answer booklet**! Your working is not marked in Part A.

Write your answers clearly. Questions are not all of equal value. Keep your answers neat and very brief. Messy or long answers will not be marked.

Question 1

(3 marks)
Write out the contents of a small file sum.c which complies with the course style guide and, when compiled and executed on a lab machine with the commands

```
gcc -Wall -Werror -o sum sum.c
./sum
```

reads in a number n and prints out the value of $1 + 2 + ... + (n - 1) + n$. You may assume that n is a positive integer less than 10,000.

Question 2

(3 marks)
Below are the binary contents of memory locations 997..1003 on Richard's laptop while a program is running:

```
 997  01100100
 998  00000010
 999  01100101
1000  00101010
1001  01100100
1002  00000001
1003  00001010
```

The currently running program was written in C and includes the following three (consecutive) lines of printfs

```
printf ("%d\n", &x);
printf ("%x\n", x);
printf ("%d\n", x);
```

The first printf prints out 1000, in the space below write what **one** of the other two printfs would print (you may select whichever you wish), and clearly mark which printf your answer corresponds to.

It will print: _____

Question 3

(3 marks)
The following values in turn

$$31, 41, 59, 26, 53, 58, 97, 93, 23, 33, 32$$

are inserted into an ordered binary tree. The tree is initially empty. In the space below clearly draw the tree after all the values have been inserted.

Question 4

(3 marks)

Three 15 bit packets have been hamming encoded with even parity and sent to you across a network. Each packet has eleven data bits and four check bits.

You receive the 3 data packets below (bits have been numbered to assist you). Interference during transmission has caused some bits to be altered (at *most* one bit in each packet luckily). Circle all the altered bits, and write "OK" next to a packet if it contains no altered bits.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	0	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1	1	0	0	1	0	1	0
0	1	0	1	0	0	0	0	0	0	0	0	0	1	0

Question 5

(4 marks)

An 8918 program starting at location 0 (shown below with correct comments) computes whether or not $x < y$. x is stored at location 40, y at 41, and the result of the program is written in location 42 using the C convention that zero means false and non-zero means true. Sadly the program has a small logic bug.

```
9 40   % R0 <- x
10 41  % R1 <- y
2      % R0 = R0 - R1
14 14  % if (R0==0) go to 14
1      % R0 = R0 + R1
5      % R0--
15 4   % if (R0 != 0) go to 4
3      % R0++
11 42  % store R0 -> cell 42
```

Briefly explain what is the problem with the program.

Question 6

(3 marks)

What does the following code print when compiled and executed?

```
#include <stdio.h>

int ten (int n);
int main (int argc, char *argv[]) {
   printf ("%d\n", ten (9876));
   return 0;
}

int ten (int n) {
   int Ten = 10;
   if (n < 10) {
      Ten = n % Ten;
   } else {
      Ten = ten ((n % Ten) + ten (n / Ten));
   }
   return Ten;
}
```

Output _____

Question 7

(3 marks)

The following code fragment contains two potentially serious problems which `gcc -Wall -Werror` will not detect. **Circle each** and give a brief correction in each case. If there are more than two problems select the two which are the most serious.

```
// ... header and more #includes and #defines above here

#define FENCE_LENGTH 20
#define NUM_POSTS    FENCE_LENGTH + 1
#define TRUE         1
#define FALSE        0

int countRotten (void) {
   int post = 0;
   int numRotten = 0;
   int rottenPosts [NUM_POSTS];
   while (post < NUM_POSTS) {
      if (isRotten (post) == TRUE) {
         rottenPosts [numRotten] = post;
         numRotten++;
      }
      post++;
   }
   // printf ("There are %d rotten fence posts\n", numRotten);
   return numRotten;
}

// more functions below here ...
```

Question 8

(3 marks) Briefly explain the meaning of the parameters argv and argc in a C main function.

argc: _____

_____

argv: _____

_____

Question 9

(3 marks)

Suppose gcc has been enhanced so it can produce 8919 machine code for the 8919 processor. On my 8919 machine the first 20 bytes of the memory currently has the contents:

```
 2  3  6  8
10 12 16  5
19  4  7 14
 6 13 18  7
 0  1 19 13
```

If x is stored in memory cell 10, what is the value of the C expressions below? (Assume x has an appropriate type in each case so that the expressions are valid C.)

`*&x` = _____

`***&x` = _____

Question 10

(3 marks)

Consider the following C program.

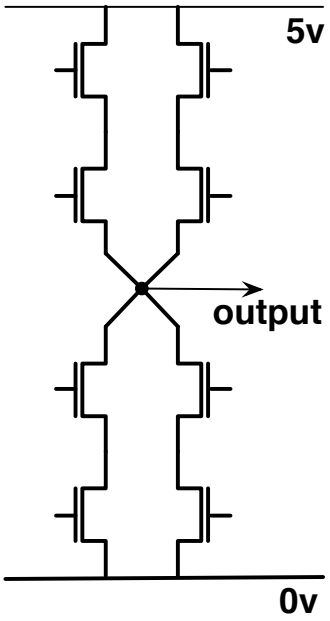
```
#include <stdio.h>
int g (int *w, int e, int *n);
int main (int argc, char** argv) {
   int d = 1;
   int o = 2;
   int t = 3;
   int *i;
   i = &d;
   printf("d=%d; o=%d; t=%d; i=%p \n", d, o, t, i);
   g (i, o, &t);
   printf("d=%d; o=%d; t=%d; i=%p \n", d, o, t, i);
   return 0;
}

int g (int *w, int e, int *n) {
   e++;
   (*n)++;
   return (*w)+1;
}
```

Which, if any, variable(s) has(have) a different value printed by the second printf? _____

Question 11

(3 marks)

The following circuit computes A *XOR* B, but some information has been omitted from the diagram. Each of the 8 transistors needs to be connected to either A, or to B, or to $\overline{A}$, or to $\overline{B}$ (recall $\overline{A}$ means in inverse of A, ie $\overline{A}$ is high when A is low, and low when A is high; and that x *XOR* y is true if x is true, or if y is true, but not if they are both true.)



Indicate by clearly marking on the diagram which of A, B, $\overline{A}$, or $\overline{B}$ each transistor is connected to.

Part B

Answer this part in your Part B answer booklet. Start each question on a new page.

Make your answers as clear and easy to understand as possible. Provide brief comments in your code where necessary. Confusing or illegible solutions will lose marks.

If you do not wish your answer for a question to be marked, clearly record 1 mark for that question on the front of your Part B answer booklet. If you do this your answer for that question will **not** be marked.

Question 12

(5 marks)

Write a C function `void wondrous (int start)` which prints out the sequence of numbers generated by the following rule: if the current term is even then the next term is half the current term, otherwise the next term is given by 3 times the current term, plus one. The function is provided the initial term to be printed and is to print terms until the value 1 is reached. You may assume the initial value is greater than 1.

For example `wondrous (10)` would print

```
10
5
16
8
4
2
1
```

Question 13

(4 marks)

State, giving an extremely concise example for each, two different types of errors detected by mudflap and/or valgrind.

Question 14

(7 marks)

Assume the machine code generated by compiling the program below on a simple 32 bit machine follows our own X19 calling convention. Write out the contents of the stack during the execution of the program, at the instant immediately before the printf is called. Show from the top of the stack down to the frame for the main function. Fill in exact (where known) or plausible (where not known) values for each word on the stack and, using arrows, briefly state what each item on the stack is and why it is needed. (You may write some or all values in decimal or as ASCII characters if you wish, there is no need to use binary or hexadecimal unless you wish to do so.)

```
#include <stdio.h>
#include <assert.h>
int factorial (int n);
int main (int argc, char *argv[]) {
   factorial (6);
   return 0;
}

int factorial (int n) {
   int result = 1;
   assert (n >= 0);
   if (n > 1) {
      result = n * factorial (n - 1);
   }
   if ((result==24) || (result==42)) {
      printf ("Hello Australia\n");
   }
   return result;
}
```

Question 15

(6 marks)

Write a function in C to compare two strings, which returns a non-zero value if the strings have the same contents, and zero otherwise.

You may not use any C library functions. Hence you can NOT use strcmp, strncmp etc.

Question 16

(8 marks)

Write a function which, given a string containing a positive integer and a word, copies the integer and the word into the fields of a struct of this type:

```
typedef struct {
int number;
char name[256];
} pair;
```

Your function must have this prototype

```
void strToCount(char *string, pair *pairPointer);
```

You may not use any C library functions. Hence you can NOT use strcpy, atoi, isdigit etc. Assume the word is a sequence of lower-case alphabetic characters (a..z). Assume the number contains only the digits (0..9). Zero or more space characters can occur before the number, between the number and the word, and after the word.

Part C

Answer this part in your Part C answer booklet. Start each question on a new page.

Make your answers as clear and easy to understand as possible. Provide type definitions and brief comments where necessary. Confusing or illegible solutions will lose marks.

If you do not wish your answer for a question to be marked, clearly record 1 mark for that question on the front of your Part C answer booklet. If you do this your answer for that question will **not** be marked.

A priority queue is a standard queue with the following difference: the elements in the queue are kept in order. So when an item is added to the queue it is not added at the tail, it is placed in the correct location depending upon its value.

For example if 6 was added to a priority queue of ints, and all the other numbers in the queue were bigger than 6, then the 6 would be placed at the head of the queue. On the other hand, if all the other numbers in the queue were smaller than 6 then it would be placed at the tail of the queue, and if the other numbers in the queue were $1, 2, 3, 10, 11, 12$ then it would be placed in the middle.

In this Part we want to define an ADT PriorityQueue, to represent a priority queue of ints, and then use that ADT to sort a list of ints.

Start each question below on a new page.

Question 17

(6 marks)

Write the file PriorityQueue.h which gives prototypes for the interface functions and defines an abstract type PriorityQueue. The interface is to contain 5 or 6 functions.

Hint: The interface functions needed are probably the same functions you would need for any queue. Also, you should keep the interface as simple and small as possible as you will have to implement it in the question below, and note that in a queue you only need to be able to delete elements from the front of the queue (easy) rather than from an arbitrary position in the queue (harder).

Question 18

(9 marks)

Write the file PriorityQueue.c which implements the PriorityQueue interface functions and any concrete types required. The queue is to be implemented using a linked list.

Question 19

(5 marks)

Write a file main.c with a main function which reads in integers one at a time using scanf until a negative number is entered, and then prints them all out in increasing order, one per line. It is to sort the numbers using the PriorityQueue ADT you have written in the questions above.

In your answer you may assume that the input will consist of only numbers and spaces.

Part D: Challenge Questions

Answer this part in your Part D answer booklet.

Partial solutions for these questions (i.e. attempts worth less than 50%) will score no marks in this part. If you do not wish your answer for a question to be marked, record 1 mark for that question on the front of your Part D answer booklet. If you do this your answer will **not** be marked.

Your solution must be clear, elegant and easy to understand. In this part confusing or difficult to understand solutions will score no marks.

Question 20

(10 marks)

For this question you are to devise an algorithm to delete a node from a sorted binary tree, BUT you want the alorithm to alter at most 3 other nodes in the process, and you do not want it to increase the depth of the tree at all (the depth of the tree means the number of layers in the tree, or in other words the length of the longest path from the root node to a leaf.)

(i) **Using a sequence of clear drawings** and brief annotations clearly show how your algorithm works.

(ii) **Write a simple implementation** of your algorithm as a C function. Indicate which parts of the function correspond to which drawings from the previous sub-question. *Take no more than one page in total to answer this sub-question.*

Question 21

Answer **only** one of the sub-questions below. In your answer clearly indicate which sub-question you are attempting. If you accidentally answer more than one clearly indicate which you want to be marked. Notice that the easier sub-questions are worth less marks.

You must (briefly) comment each line of machine code you write in this question, stating the instruction you are executing and a brief indication of what the line is achieving.

In all the sub-questions store the number n in the highest numbered memory address (4 or 8 bit chips) or pair of memory addresses (16 bit chip) of the microprocessor that you are using (eg in the 4917 it would be stored in the cell at location 15).

(i) *(8 marks)* Write a program for one of our 8 or 16 bit microprocessors (you can choose which) to sum the numbers $n..0$ using recursion. Implement a recursive function in your solution, which uses stack frames in its operation. Use our X19 calling convention. Clearly state which microprocessor you have used and why. Remember to comment your answer carefully - your code needs to be very clear and clean.

(ii) *(4 marks)* Write an 8918 program to sum the numbers $n..0$ using a loop.

(iii) *(2 marks)* Write a 4917 program to print out the numbers $n..0$ using a loop.