Family Name:	
Other Names:	
Signature:	
Student Number:	

This PAPER is NOT to be retained by the STUDENT

The University Of New South Wales

C0MP1917 Final Exam Public—Morning Session

Computing 1

June 2010

Time allowed: **3 Hours**
Total number of questions: **23**
Total number of marks: **100**

Your answers must be submitted electronically using the `submit` command as described in each part.
You may submit your solutions as many times as you like. The last submission **ONLY** will be marked.
Answers to questions in PART A, PART B and PART C must be answered in the appropriate place in the provided file `Answers.txt`. This file can be submitted using the command:

```
        submit written Answers.txt
```

Answers to questions in PART D and PART E must be answered in a file named as specified in the question and submitted using the `submit` command as specified at the end of the question.
You can check what you have submitted by using the `submit` command without arguments.

No examination materials permitted.
Calculators may **not** be used.
Questions are **not** worth equal marks.
Answer **all** questions.

Examiner's Use Only:

Inst	Part A	Part B	Part C	Part D	Part E	**Total**

Part A: Short Answer Questions

NOTE: Answer the questions in this section in the file `Answers.txt` provided.

Note that each question has *five* alternatives. Once you have chosen an alternative, un-comment the appropriate alternative in the file `Answers.txt` by deleting the two percent symbols (i.e., `%%`) at the start of the line. Un-comment one alternative only.

Each question in this section is worth **2 marks**. There is a **penalty** of $-\frac{1}{2}$ **mark** for answering a question in this section incorrectly. There is no penalty for not answering a question. In other words, you get no marks for a question if you do not attempt it and you lose half a mark for getting a question wrong.

Question 1

Consider the following `for` loop:

```
while (i > 1) {
    printf("Hello world!\n");
    i--;
}
```

What value should `g` be given so that the string "`Hello world!\n`" is printed exactly 10 times?

[A] 10
[B] 11
[C] 12
[D] 13
[E] None of the above

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 2

Given these declarations:

```
int x = 7;
double y = 2;
```

What type is this C expression: `(x / y + 2)`

[A] `double`
[B] `long`
[C] `int`
[D] `String`
[E] `byte`

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 3

Given a binary search tree, which of the following methods for traversing the tree will give the best indication of the order in which the nodes were added to the tree? NO LONGER COVERED IN COMPUTING 1

`[A]` prefix traversal
`[B]` infix traversal
`[C]` postfix traversal
`[D]` suffix traversal
`[E]` none of the above

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 4

Consider the following C definition:

```
    char str[] = "Hello World!";
```

The size of the array `str` in characters is:

`[A]` 12
`[B]` 13
`[C]` 24
`[D]` 26
`[E]` 52

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 5

Given the following C program:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 4

int main (int argc, char *argv[]) {
    int i, j = 0;

    for(i = j; i < MAX * 2; i++) {
        }
        printf("i = %d\n", i);
        j += i;
    printf("j = %d\n", j);
    return(EXIT_SUCCESS);
}
```

What value is printed for the variable `j` at the end of the program?

```
[A]  0
[B]  1
[C]  4
[D]  8
[E]  28
```

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 6

Which of the following `while` loops gives the same final value for the variable `sum` as this `for` loop:

```
        sum = 0; for (i = 2; i < 79; i++) {sum += i;}
```

```
[A]  sum = 0; i = 2; while (i <= 79) { sum += i; i++; }
[B]  sum = 0; i = 1; while (i < 79) { sum += i; i++; }
[C]  sum = 0; i = 2; while (i < 80) { sum += i; i++; }
[D]  sum = 0; i = 1; while (i < 78) { i++; sum += i; }
[E]  sum = 0; i = 1; while (i < 79) { i++; sum += i; }
```

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 7

Which of the following MIPS instructions would be used to place a constant integer value in a register?

```
[A]  add
[B]  la
[C]  li
[D]  move
[E]  sub
```

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 8

The following definitions are used to represent a stack as a linked list of items. The variable `stack` points to the top of the stack.

```
typedef struct listNode ListNode;

struct listNode {
    int item;
    ListNode *next;
};

ListNode *stack = NULL;
ListNode *tmp;
```

Assume one or more items have been "pushed" onto the stack.

Which one of the following piece of code correctly "pops" the first element from the top of the stack and deallocates any now unused memory?

[A]

```
        tmp = stack;
        tmp = tmp->next;
        free(stack);
```

[B]

```
        tmp = stack;
        free(tmp);
        stack = stack->next;
```

[C]

```
        stack = stack->next;
        free(stack);
```

[D]

```
        tmp = stack;
        stack = tmp->next;
        free(tmp);
```

[E]

```
        tmp = stack;
        stack = tmp->next;
        free(stack);
```

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 9

Consider the following C program:

```
#include <stdio.h>

int main (int argc, char *argv[]) {
    int q;
    int *p = NULL;

    q = 24;
    *p = 42;
    q = *p;
    printf("The value of q is %d\n", q);

    return (EXIT_SUCCESS);
}
```

What value is printed for the variable `q`?

`[A]` 0
`[B]` 24
`[C]` 42
`[D]` 1008
`[E]` undefined - the program is invalid

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Question 10

Consider the following C function prototype:

```
int f(int &x, int y);
```

Which of the following statements is true of the argument `x`?

`[A]` It is passed by argument.
`[B]` It is passed by difference.
`[C]` It is passed by reference.
`[D]` It is passed by value.
`[E]` None of the above.

Record your answer by uncommenting your choice in the file `Answers.txt`. Submit your answers using the command:

```
submit written Answers.txt
```

Part B: C Program Understanding

NOTE: Answer the questions in this section in the file `Answers.txt` provided.

Make your answers as clear and easy to understand as possible. Confusing or illegible solutions will lose marks.

Question 11

(5 marks)

Consider the following valid C program.

```
#include <stdio.h>
#include <stdlib.h>

void f (int *x, int *y);

int main (int argc, char *argv[]) {
    int a = 2, b = 1, c = 3;

    printf ("a = %d b = %d c = %d\n", a, b, c);
    f (&a, &b);
    f (&b, &c);
    f (&a, &b);
    printf ("a = %d b = %d c = %d\n", a, b, c);

    return EXIT_SUCCESS;
}

void f(int *x, int *y) {
    if (*x > *y) {
        int temp = *x;
        *x = *y;
        *y = temp;
    }
}
```

[A] Briefly state the purpose of the function `f`.

[B] Briefly state the purpose of the entire program.

[C] Indicate clearly and exactly what output will be printed when the program above is compiled and run.

Question 12

(6 marks)

Consider the following C program for determining the moves required to solve the *Towers of Hanoi* problem.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void hanoi(int n, char from, char to, char spare);

int main (int argc, char *argv[]) {

    printf("Solution of Towers of Hanoi problem with 3 discs\n");
    hanoi(2, 'A', 'B', 'C');

    return EXIT_SUCCESS;
}

void hanoi (int n, char from, char to, char spare) {
    if (n > 0 ) {
        hanoi (n - 1, from, spare, to);
        printf ("Move %c to %c\n", from, to);
        hanoi (n - 1, spare, to, from);
    }
}
```

Write down the sequence of calls to the function `hanoi`, in the order that they are called, when this program is run and the values of the arguments for each call. For example, the first function call is `hanoi(2, 'A', 'B', 'C')`.

Question 13

(6 marks)

Consider the memory allocation:

```
int *a = (int *)malloc(15 * sizeof(int));
```

[A] How many bytes of memory will be allocated to `a` assuming an integer occupies 4 bytes?

[B] If the return value from `malloc()` is `NULL`, what does this indicate?

[C] If the return value from `malloc()` is `0x803B000` (in Hexadecimal notation), what will be the address of `a[1]`, `a[5]` and `a[15]` (also in Hexadecimal notation)?

[D] What will be printed when this additional code is executed?

```
int i;
for (i = 0; i < 4; i++) {
    a++;
    a[i] = i;
}
printf("%d\n", a[1]);
```

Question 14

(6 marks)

After a long night of watching World Cup matches on TV, the chief programmer for XYZ Pty Ltd has written the following implementation of the `Node *pushLast(Node *newItem, Node *list)` function whose purpose is to add `newItem` to the linked list pointed to by `list` and return the new first element of the linked list. (The numbers on the left-hand side are line numbers.)

```
1  typedef struct node Node;
2
3  struct node {
4      char data;
5      Node *next;
6  };
7
...

8  Node *pushLast(Node *newItem, Node *list) {
9      Node *returnNode;
10
11     if (list == NULL) {
12         returnNode = NULL;
13     } else {
14         for (tmp = list; tmp != NULL; tmp = tmp->data) {
15         }
16         tmp->next = newItem;
17         returnNode = list;
18     }
19     return(returnNode);
20 }
```

Unfortunately they have made exactly 4 errors in the implementation of the `pushLast` function. Identify each error by stating the line number on which it occurs and provide the correct C code.

Submit your answers to this part using the command:

```
submit written Answers.txt
```

Part C: Systems Understanding

NOTE: Answer the questions in this section in the file `Answers.txt` provided.

Make your answers as clear and easy to understand as possible. Confusing or illegible solutions will lose marks.

Question 15

(5 marks)

[A] Convert the following binary number to decimal.

101011011

[B] Convert the following from decimal to binary.

123625

[C] Assuming that integers are stored in 4 bits, what value does the bit pattern 1101 represent in each of the following formats? NO LONGER COVERED IN COMPUTING 1

i) Unsigned integer

ii) Sign-magnitude

iii) Two's complement

Question 16

(5 marks)

Consider the following C program. It is correct and runs without error.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define LIMIT   5

void fun(int i1, char *s1, int i2, char *s2);

int main (int argc, char *argv[]) {
    int total = 0;
    int items = 0;
    char itemStr[] = "Number of items = ";
    char totalStr[] = "total =";

    for (items = 0; items < LIMIT; items++) {
        total += items;
    }

    fun (total, totalStr, items, itemStr);

    return EXIT_SUCCESS;
}

void fun (int i1, char *s1, int i2, char *s2) {
    int avg = i1/i2;
    static char str[] = "Average = ";

    printf ("%s %d; %s %d ", s1, i1, s2, i2);
    printf ("%s %d", str, avg);
}
```

For each of the program elements below indicate where in memory they would likely be stored in a CSE lab computer's memory (e.g., stack, heap, etc.) and whether the address in memory is likely to be a high address (e.g., 0xFFFFFF) or a low address (e.g., 0x000000).

[A] The variable `total` in the function `main()`.

[B] The variable `itemStr` in the function `main()`.

[C] The variable `avg` in the function `fun()`.

[D] The parameter `i1` in the function `fun()`.

[E] The compiled code for the function `main()`.

Submit your answers to this part using the command:

```
submit written Answers.txt
```

Part D: C Programming

Answer this part in files named as specified in each question.
Submit your answers using the command specified for each each question.
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.
Unless otherwise specified each program will be compiled using the following command:

```
gcc -Wall -Werror -O -o file file.c
```

where `file.c` is the name of the file in which your answer is placed as specified in the question.

Question 17

(5 marks)

Write a complete C program called `grid.c` that takes a defined constant `MAXNUM` and prints a `MAXNUM` × `MAXNUM` grid using the ASCII characters `+` (plus) and `-` (minus). Using these two symbols a grid cell can be constructed by arranging the characters as follows:

```
+-+
+-+
```

For example, with `MAXNUM` given the value 3

```
    % ./grid
    Grid size: 3
    +-+-+-+
    +-+-+-+
    +-+-+-+
    +-+-+-+
```

You may use the following C program skeleton (provided in file `grid.c.skel`):

```
#include <stdio.h>
#include <stdlib.h>

// YOU MAY ALTER THE VALUE OF THE FOLLOWING CONSTANT BUT NOT ITS NAME
#define MAXNUM  10

int main (int argc, char *argv[]) {
    // YOUR CODE GOES HERE

    printf ("Grid size: %d\n", MAXNUM);

    // YOUR CODE GOES HERE

    return EXIT_SUCCESS;
}
```

Submit your answer using the command:

```
submit 17 grid.c
```

Question 18

(6 marks)

Write a complete C program called `rainfall.c` that reads in a sequence of daily rainfall figures (measured in integral millimetres) and computes the number of days on which rain data was recorded, the total rainfall, the average rainfall correct to two decimal places and the number of "wet days". A "wet day" is defined to be any day where more than 5mm of rain falls.
Your program should accept a sequence of integers on standard input until the end-of-file (EOF) character is received (identified by `^D` (control-D) at the start of a line in the example below).
Make your program behave exactly as indicated by the examples below.
It must produce exactly the same output as below.
For example:

```
% ./rainfall
Enter rainfall: 5 6 4 0 9
15 10 7 4 0 0 1
^D
Days: 12 Total rainfall: 61mm Average rainfall: 5.08mm Wet days: 5
```

Submit your answer using the command:

```
submit 18 rainfall.c
```

Question 19

(7 marks)

An anagram is a word or phrase obtained by rearranging the letters of another word or phrase. For example:

> "neo" is an anagram of "one"

Write a complete C program called `anagram.c` that prompts for two strings *string1* and *string2* and then prints a message saying whether *string1* is an anagram of *string2*.
For the purposes of this task, you may assume that *string1* and *string2* consist of alphabetic and numeric characters only.
Your program should be capable of dealing with strings consisting of 100 characters.
Make your program behave exactly as indicated by the examples below.
It must produce exactly the same output as below. For example:

```
% ./anagram
Enter string1: neo
Enter string2: one
"neo" is an anagram of "one"
% ./anagram
Enter string1: one
Enter string2: blob
"one" is not an anagram of "blob"
% ./anagram
Enter string1: clinteastwood
Enter string2: oldwestaction
"clinteastwood" is an anagram of "oldwestaction"
```

Submit your answer using the command:

```
submit 19 anagram.c
```

Question 20

(7 marks)

Write a C program `sign.c`.

A company that produces neon signs has asked you to write a program that will determine the cost of signs that it will produce for its customers and identify the dearest sign. The cost of a sign is determined by the cost of producing each individual character in the sign. The cost of individual characters is determined as follows:

characters	value
`a/A`	\$1
`b/B`	\$2
`c/C`	\$3
...	...
`z/Z`	\$26
`0 – 9`	\$5
all other characters	\$2

Write a complete C program `sign.c` which reads in a sequence of strings up until the typing of the control-D character (indicated by `^D` below) and then outputs the dearest string of those entered together with its cost.
A string is any sequence of non-space characters.
Make your program behave **exactly** as indicated by the examples below.
Your program should be able to deal with 100 words of input which are a maximum of 100 characters long.
If more than one string has the same cost as the dearest string, the first that appears on the input should be returned.
It must produce **exactly** the same output as below.

```
% ./sign
Enter text:
number1 hello!!!!
this_is-a_long-sign
^D
Dearest sign is this_is-a_long-sign with cost $190
```

Submit your answer using the command:

```
submit 20 sign.c
```

Question 21

(12 marks)

Given a list of elements, the list can be *rotated to the left* by moving the first element of the list to the back of the list. For example, the list 1, 3, 5, 7, 9, 11 can be rotated one place to the left producing the list 3, 5, 7, 9, 11, 1. This process can be repeated to rotate the list through successive places. For example, the list 1, 3, 5, 7, 9, 11 can be rotated three places to the left producing the list 7, 9, 11, 1, 3, 5.
Write a C function `Node *rotate(Node *first, int n)` which, given a non-empty linked list of integers (`first` points to the first element of the list) and an integer `n`, returns the linked list rotated `n` places to the left.
Each element of the linked list is represented with this struct:

```
typedef struct node Node;

struct node {
       int        data;
       struct node *next;
};
```

Your function **rotate** must have this prototype:

```
Node *rotate(Node *first, int n);
```

Your file **MUST** contain a definition of `struct node` and `Node` as above.
You are not permitted to use arrays anywhere in your answer to this question.
You are not permitted to use any functions from the C library in answering this question.
You may write extra functions, if you wish, in answering this question.
You can assume the list has at least one item in it. You can assume that the argument `n` is greater than 0.
You are only required to supply the function `rotate`. A skeleton is provided in the file `rotate.c.skel` if you wish to use it.
No error checking is required.
A file named `q21_rotate.c` is provided to help you test your code. When compiled with the following command

```
gcc -Wall -Werror -O -o q21 q21_rotate.c rotate.c
```

(where `rotate.c` contains your code) the program takes a sequence of integers on the command line. The first integer specifies the amount of rotation and the remaining integers constitute the elements of the list. For example, the following example rotates the list 1, 2, 3, 4, 5 by 3 places.

```
% ./q21 3 1 2 3 4 5
Original list:  1 2 3 4 5
List rotated 3:  4 5 1 2 3
```

Your file `rotate.c` that you submit must not contain a `main()` function.
Submit your answer using the command:

```
submit 21 rotate.c
```

Part E: Systems Programming

Answer this part in files named as specified in each question.
Submit your answers using the command specified for each each question.
Make your answers as clear and easy to understand as possible. Confusing or illegible solutions will lose marks.
Your solutions must be clear, elegant and easy to understand.

Question 22

(3 marks)

Write a MIPS assembler program `oddeven.s` which:

- prompts for and reads one integer n
- the prompt should read `Enter a number:`
- if n is odd, prints the string “Number is odd”
- if n is even, prints the string “Number is even”

A program skeleton is provided in the file `oddeven.s.skel` if you wish to use it.
You may assume that the input to your program is given exactly as one integer > 0.
No error checking is necessary.
Make your program behave **exactly** as indicated by the examples below.
It must produce **exactly** the same output as below in the `xspim` console window.
In these examples, the program has been executed 3 times. The input and results are shown in each case. Note that blank lines are not required in your output.

```
% spim -f oddeven.s
Enter a number: 5
Number is odd

% spim -f oddeven.s
Enter a number: 2
Number is even

% spim -f oddeven.s
Enter a number: 258
Number is even
```

Submit your answer using the command:

```
submit 22 oddeven.s
```

Question 23

(7 marks)

Write a MIPS assembler program `sumdigits.s` which:

- prompts for and reads one integer n
- the prompt should read `Enter a number:`
- output the sum of the individual digits of n

You may assume that the input to your program is given exactly as one integer > 0.
No error checking is necessary.
Make your program behave **exactly** as indicated by the examples below.
It must produce **exactly** the same output as below in the `xspim` console window.
In these examples, the program has been executed 3 times. The input and results are shown in each case. Note that blank lines are not required in your output.

```
% spim -f sumdigits.s
Enter a number: 1
Sum of digits is 1

% spim -f sumdigits.s
Enter a number: 123
Sum of digits is 6

% spim -f sumdigits.s
Enter a number: 1234
Sum of digits is 10
```

Submit your answer using the command:

```
submit 23 sumdigits.s
```