Family Name: .....

Other Names: .....

Signature: .....

Student Number: .....

THE UNIVERSITY OF NEW SOUTH WALES
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Sample Examination

COMP1917 Higher Computing 1

Exam Duration: 3 hours **Total marks for this paper: 100**

This paper has 14 pages including this cover page.

Authorised materials:

- One hand-written A4 sheet of paper (double sided).

Instructions to Invigilators:

- Please provide each student with one 24-page Script Book.
- Please **collect** question papers and A4 sheets.

Instructions to Students:

- This paper counts for 60% of your final grade.
- Answer in the **Script Books** provided.
- Hand in question paper and A4 sheet when you are finished.
- All questions may be attempted.
- Start your answer to each question on a new page.

Note: you are **not** required to provide header documentation for any of the functions or programs in this exam. Comments are only required to the extent that they will help the marker understand the structure of your code.

Question 1.

a. (6 marks) Write a program to read 17 integers from standard input and print the minimum, maximum and sum of these integers to standard output.

b. (6 marks) An array pointed to by the pointer variable `p` has `n` values of type `int` stored in it.

Write a **function** whose prototype is:

```
int replace_val(int *p, int n, int old_val, int new_val);
```

which replaces in the array the first occurrence of the number whose value is `old_val` with `new_val`. If the item `old_val` is not found in the array, nothing is replaced. The function returns the index in the array at which the replacement has taken place, or -1 if nothing has been replaced.

c. (5 marks) Write a program to (i) read characters from standard input into an array of char until 500 characters have been read, or `EOF` is reached; (ii) print these characters to standard output in the reverse of their original order.

d. (3 marks) Write a function to return the length of a string, without using library functions.

Question 2.

a. (4 marks)

Consider this C program:

```
#include <stdio.h>

int main( void )
{
    int x=1;
    int y=0;

    while( x < 100 ) {
        x = x + 8 * y++ ;
        printf("%d\n", x );
    }

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.

2 b. (4 marks)

Consider this C program:

```
#include <stdio.h>

int main( void )
{
    int *p, *q;
    int  x, y;

    x = 7;
    y = 8;

    q = &x;
   *q = 10;
    p = q;

    y = *p + *q;
   *p =  x + y;

    printf("x = %d, y = %d\n", x, y );

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.

2 c. (4 marks)

Consider this C program:

```
#include <stdio.h>


int df( int n )
{
    if( n < 2 ) {
        return( 1 );
    }
    else {
        return( n * df( n-2 ));
    }
}

int main( void )
{
    printf( "5!! = %d\n", df( 5 ));
    printf( "6!! = %d\n", df( 6 ));

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.

2 d. (4 marks)

Consider this C program:

```
#include <stdio.h>

void print_edge( int k )
{
    int i;

    for( i=0; i < k; i++ ) {
        printf( "+-" );
    }
    printf( "+\n" );
}

void print_legs( int k )
{
    int i;

    for( i=0; i < k; i++ ) {
        printf( "| " );
    }
    printf( "|\n" );
}

int main( void )
{
    int k;

    for( k=0; k < 4; k++ ) {
        print_edge( k );
        print_legs( k );
    }
    print_edge( k );

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.

2 e. (4 marks)

Consider this C program:

```
#include <stdio.h>

void  merge( int a[], int r, int b[], int s, int c[] )
{
    int i=0, j=0, k=0;

    while(( i < r )&&( j < s )) {
        if( a[i] < b[j] )
            c[k++] = a[i++];
        else
            c[k++] = b[j++];
    }
    while( i < r )
        c[k++] = a[i++];
    while( j < s )
        c[k++] = b[j++];
}

int main()
{
    int c[9];
    int a[] = { 6, 5, 2, 8, 3, 1, 4, 9, 7 };
    int i;

    merge( a, 4, &a[4], 4, c );

    for( i=0; i < 8; i++ ) {
        printf( " %d", c[i] );
    }
    printf( "\n" );

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.

Question 3.

a. (6 marks) Write a function `test_sort()` whose prototype is

```
int test_sort( int nPtr[], int n );
```

which takes an array `nPtr[]` of `n` integers, and returns 1 or 0 as follows:

1	if the integers are sorted in strictly **increasing** order,
1	if the integers are sorted in strictly **decreasing** order, and
0	if they are not in sorted order, or if two or more of them are equal.

b. (4 marks) Consider this C program:

```
#include <stdio.h>

void f( char *s )
{
    if( !*s ) {
        return;
    }
    f( s+1 );
    putchar( *s );
}

int main(void)
{
    f("kernighan");
    putchar('\n');

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.

3 c. (5 marks) Consider this C program:

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

struct node {
    int value;
    struct node *next;
};

struct node *new( int val )
{
    struct node *n = malloc(sizeof(struct node));
    n->value = val;
    n->next  = NULL;
    return n;
}

struct node *insert( struct node *n, struct node *h )
{
    n->next = h;
    return n;
}

int main( void )
{
    struct node *a, *b, *c;
    int i;

    a = new(1);
    b = insert( new(5), insert( new(2), a ));
    c = insert( new(3), b->next );
    a = insert( a, c );

    for( i=0; i < 5; i++ ) {
        printf(" %d", b->value );
        b = b->next;
    }
    printf("\n");

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.

3 d. (5 marks) Consider this C program:

```
#include <stdio.h>

int a = 1;
static int b = 1;

int f( int c )
{
    static int d = 1;
    int e = 0;

    a++;
    b += d;
    c = c + 2;
    d = d + a - b + c;
    e = e + 2*d + 1;
    return( e+2 );
}

int main( void )
{
    int a, d;
    a = 3;

    for( d=0; d < 3; d++ ) {
        printf("%d\n", f(a));
    }
    printf("%d\n", a );
    printf("%d\n", b );
    printf("%d\n", d );

    return 0;
}
```

The program is valid C. It executes without error.
Indicate clearly and exactly what output will be printed.

Question 4.

a. (9 marks) Write a program to print to standard output the number of **characters**, the number of **words**, and the number of **lines**, in a text file whose name is provided in the command-line call to the program. [For this question, a "word" is a seqence of characters for which the library function `isspace()` returns 0].

b. (11 marks) The following program is designed to read a list of stock market quotes and print them to standard output in order of (decreasing) price. Each quote consists of a price per share followed by a company name. For example, the input

```
$14.40      AMP
$24.35      CBA
$17.99      BHP
```

should produce the output

```
$24.35      CBA
$17.99      BHP
$14.40      AMP
```

The relevant structures and function prototypes are given below. The coding for `main()` and `get_quote()` are provided on page 5. You are to supply the code for the functions `list_insert()`, `print_quotes()` and `free_listmem()`.

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

#define MAX_NAME 20

typedef struct listnode Lnode;

struct listnode {
  char   name[MAX_NAME];
  float  price;
  Lnode *next;
};

Lnode *get_quote( void );
Lnode *list_insert( Lnode *head, Lnode *newn );
void  print_quotes( Lnode *curr );
void  free_listmem( Lnode *head );
```

```
int main( void ) {

  Lnode *list = NULL, *newn;

  while (( newn = get_quote() ) != NULL ) {
    list = list_insert( list, newn );
  }
  print_quotes( list );
  free_listmem( list );

  return 0;
}

Lnode *get_quote( void )
{
  Lnode *newn;

  newn = (Lnode *) malloc( sizeof( Lnode ));
  if ( newn == NULL ) {
    printf( "Memory allocation failure!\n" );
    exit( EXIT_FAILURE );
  }
  newn->next = NULL;
  if ( ( scanf( "$%f", &newn->price ) == 1)
     && fgets( newn->name, MAX_NAME, stdin )) {
    return newn;
  }
  else {
    return NULL;
  }
}

/* PROVIDE CODE FOR list_insert()  */

/* PROVIDE CODE FOR print_quotes() */

/* PROVIDE CODE FOR free_listmem() */
```

Question 5.

a. (3 marks) Convert the following binary number to decimal.

10101.001_2

b. (2 marks) Convert the following decimal number to binary.

45.375_{10}

c. (5 marks) Assume that signed binary numbers are stored in two's complement form, in 8 bits (for example, $11111111_2 = -1_{10}$).

(i) Convert the following binary number to decimal.

01101011_2

(ii) What is the negative of the number from part **c(i)**, written in two's complement (binary) form?

(iii) Use your answer from part **c(ii)** to compute the result of the following binary subtraction. Write your answer in two's complement form.

$00110001_2 - 01101011_2$

The instruction set for the simple machine presented in lectures is given in the table on page 8. The following three sub-questions refer to the program shown above the table (also on page 8). All addresses and memory contents are given in hexadecimal (base 16) notation.

d. (4 marks) What would be in registers R1, R2, R3 and R4 after running the program starting at address `A0`, given that the relevant memory contents are as shown?

e. (3 marks) Suppose that the contents of memory location `81` is changed from `02` to `03`. What would now be in register R3 after running the above program starting at address `A0`?

f. (3 marks) Write a segment of C code that accomplishes the same task as this machine language program.

Machine Language Program

```
Address  Contents

   80        00
   81        02

    .        . .

   A0       20 00
   A2       21 01
   A4       12 81
   A6       23 01
   A8       B2 B4
   AA       50 01
   AC       40 34
   AE       53 34
   B0       53 34
   B2       B0 A8
   B4       33 80
   B6       C0 00
```

Opcode	Operand	Description
1	RXY	LOAD register R with bit pattern found in memory cell whose address is XY
2	RXY	LOAD register R with the bit pattern XY
3	RXY	STORE the bit pattern found in register R in the memory cell whose address is XY
4	0RS	MOVE the bit pattern found in register R to register S
5	RST	ADD the bit patterns in registers S and T as though they were two's complement representations, and leave the result in register R
6	RST	ADD the bit patterns in registers S and T as though they represented values in floating point notation, and leave the floating point result in register R
7	RST	OR the bit patterns in registers S and T and place the result in register R
8	RST	AND the bit patterns in registers S and T and place the result in register R
9	RST	EXCLUSIVE-OR the bit patterns in registers S and T and place the result in register R
A	R0X	ROTATE the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low order end to the high order end.
B	RXY	JUMP to the instruction located in the memory cell at the address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution.
C	000	HALT execution