TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Student ID: ....

EXAMINATIONS — 2010

MID-YEAR

NWEN 241

Systems Programming

Time Allowed: 3 hours

Instructions:

- Write your student ID number at the top of each sheet.
- There are 100 possible marks on the exam.
- There are 8 questions.
- Attempt all questions.
- Make sure your answers are clear and to the point.
- Non-programmable calculators without full alphabetic keys are permitted.
- Non-electronic foreign language dictionaries are permitted.
- Refer to the Appendix.
- No other reference material is allowed.
- Answer in the appropriate heavily outlined boxes or follow the instructions given in the questions.

Question	Mark
1	
2	
3	
4	
5	
6	
7	
8	
Total	

continued...

Student ID: ....

Question 1. Shell Scripting

[15 marks]

(a) [3 marks] Consider this **sh** code fragment:

```
FIRSTNAME='John'
LASTNAME='Smith'
NAME1='$FIRSTNAME $LASTNAME'
NAME2="$FIRSTNAME $LASTNAME"
echo $NAME1
echo $NAME2
```

What would the output be and why?

(Question 1 continued on next page)

(Question 1 continued)

```
#
# [Comment 4]
#
SENDER='grep $USER /etc/passwd | cut -d: -f 5| sed -e 's/,.*//''
```

(iv) [2 marks] Write out `Comment 4`.

```
#
# [Comment 5]
#
if [ -z "$DEST" ]
then
    printf "Destination: "
    read DEST
fi

if [ -z "$TEXT" ]; then
   printf "Text: "
   read TEXT
fi
```

(v) [1 mark] Write out `Comment 5`.

```
#
# [Comment 6]
#
MESSAGE_SIZE='cho "From $SENDER: $TEXT" | wc -c'
```

(vi) [1 mark] Write out `Comment 6`.

(Question 1 continued on next page)

(Question 1 continued)

```
#
# [Comment 7]
#
if [ $MESSAGE_SIZE -le 160 ]
then
```

(vii) [1 mark] Write out `Comment 7`.

```
    #
    # [Comment 8]
    #
    ssh $SMSGATEWAY "sendsms $DEST \"From $SENDER: $TEXT\"" > /dev/null 2>&1
```

(viii) [2 marks] Write out `Comment 8`.

```
    #
    # [Comment 9]
    #
    if [ $? -ne 0 ]; then
        echo "$PROG: SMS not sent - could not contact gateway"
    fi
else
```

(ix) [1 mark] Write out `Comment 9`.

```
    #
    # [Comment 10]
    #
   echo "$PROG: SMS not sent - message is > 160"
fi
```

(x) [1 mark] Write out `Comment 10`.

continued...

Student ID: ....

Question 2. Assembly Programming [25 marks]

(a) [5 marks] Give the hexadecimal 32-bit two's complement representation of the following decimal integers, and show the details of your work:

1. 783
2. -1
3. -1321
4. -32
5. 421

(Question 2 continued on next page)

(Question 2 continued)

(b)

(i) [5 marks] Consider the following .data segment:

```
L1 dw 435
L2 db "h","e","l","l","o",0
L3 db 0A1h, 0B2h, 0C3h
L4 dw 23o
```

Show the contents of the 13 memory bytes starting at address L1, on a machine using Little Endian format. (Intel-based processors store data in memory in Little Endian format, i.e., the least significant byte is stored first. For example, the hexadecimal number 12345678 is stored in memory as 78 56 34 12.)

(Question 2 continued on next page)

(Question 2 continued)

(ii) [15 marks] Consider the following program fragment:

```
mov eax, [L3]
inc eax
mov [L2], eax
mov bx, [L1]
mov eax, L3
inc eax
mov [eax], bx
```

After the code finishes executing, what are the contents of the 13 memory bytes starting at address L1? Show your work.

Student ID: ....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

Student ID: ....

Question 3. C Basics 1 [10 marks]

(a) [2 marks] See the following two statements. Give the values of the expressions in the box below.

```
int a[3] = {11, 22, 33};
int *pa = a;
```

```
*a =

*(a+2) =

*pa =

pa[1] =
```

(b) [3 marks] See the following two statements. Give the values of the expressions in the box below.

```
int m[4][4] = {{1,3,5,7}, {11,33,55,77}, {2,4,6,8}, {22,44,66,88}};
int (*parr)[4] = m;
```

```
**m =

*(*m+2) =

*(m[1]+2) =

(*(parr+3))[2] =
```

(Question 3 continued on next page)

(Question 3 continued)

(c) [3 marks] Suppose you are working on a 32-bit machine where the size of an `int` is FOUR bytes, the size of a `char` is ONE byte and the size of pointers is FOUR bytes. See the following statements. Give the values of the expressions in the box below.

```
char *pa[] = {"12", "34", "56"};
int m[2][3] = {{1, 2, 3}, {4, 5, 6}};
int (*ppm)[2][3] = &m;
```

```
sizeof(pa) =


sizeof(**pa) =


sizeof(ppm) =


sizeof(**ppm) =
```

(Question 3 continued on next page)

Student ID: ....

(Question 3 continued)

(d) [2 marks] The following program is problematic. In the box below, explain why it is problematic. The program uses `strcpy` and `strcat`, defined in `string.h`.

`strcpy(dst, src)` copies the string `src` to `dst` (including the terminating `'\0'` character).

`strcat(s1, s2)` concatenates the strings `s1` and `s2` - a copy of `s2` is appended to the end of `s1`.

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

#define SIZE 5

int main() {
  char a1[] = "ABCD", a2[] = "abcdef", a3[] = "12345";

  strcpy(a1, a2);
  strcat(a2, a3);

  return 0;
}
```

Student ID: ....

Question 4. C Basics 2 [10 marks]

(a) See the following statements and declarations. In the boxes below, answer the corresponding questions.

```
#define mPchar char *
typedef char *tPchar;

mPchar ma, mb;
const mPchar mc, md;
tPchar ta, tb;
const tPchar tc, td;
```

(i) [1 mark] What are `ma` and `mb`?

(ii) [1 mark] What are `mc` and `md`?

(iii) [1 mark] What are `ta` and `tb`?

(iv) [1 mark] What are `tc` and `td`?

(Question 4 continued on next page)

(Question 4 continued)

(b) The following statements describe the declarations of variable p. In the boxes below, give the corresponding declarations.

For example: p is a pointer to an int.

```
int *p;
```

(i) [1 mark] p is a 5-element array of pointers to char.

(ii) [1 mark] p is a pointer to a 10-element char array.

(iii) [1 mark] p is a function that takes an int argument and returns a pointer to char.

(iv) [1 mark] p is a function that takes a char array and returns a pointer to int.

(v) [1 mark] p is a pointer to a function that returns a pointer to an int.

(vi) [1 mark] p is a function that returns a pointer to a function that returns a pointer to a 10-element int array.

Student ID: ....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

continued...

Student ID: ....

Question 5. Arrays and Pointers [10 marks]

(a) [4 marks] You have the following two char variables, c1 and c2.

```
char c1 = 'A', c2 = 'Z';
```

In the box below write a function swap that can swap the values of c1 and c2. After the swap, c1's value is 'Z' and c2's value is 'A'.

(Question 5 continued on next page)

(Question 5 continued)

(b) [4 marks] In the box below write a function `reverse` that will reverse the characters in a string. For example, the output of the following `main` function

```
main()
{ char str[] = "ABCDEFG";
  reverse(str);
  printf("%s \n", str);

  return 0;
}
```

should look like this:

```
GFEDCBA
```

You may use the string-handling function `strlen`, but NOT other string-handling functions. `strlen(s)` returns the number of characters in string `s`, not counting the terminating NULL character.

(Question 5 continued on next page)

Student ID: ....

(Question 5 continued)

(c) [2 marks] In the box below write a `main` function that will echo the command line arguements (that is, you need to implement a version of the `echo` command).

Student ID: ....

Question 6. Dynamic Data Structures and Iteration vs. Recursion [10 marks]

In this question, you need to implement functions that create a singly-linked list using iteration and recursion. You need to use the following type definitions, macro definitions and function prototypes to implement your functions:

```
#define Node_Size sizeof(Node)

typedef struct node
{ char data;
  struct node *next;
} Node;

typedef Node *ptrNode;

ptrNode createlisti(char *);    /* iteration */
ptrNode createlistr(char *);    /* recursion */
```

(a) [4 marks] Implement the function `createlisti` using iteration. The function will create a list with a character per node from a string, and return a pointer to the head of the resulting list.

(Question 6 continued on next page)

Student ID: ....

(Question 6 continued)

(b) [4 marks] Implement the function `createlistr` using recursion. The function will create a list from a string, and return a pointer to the head of the resulting list.

(Question 6 continued on next page)

Student ID: ....

(Question 6 continued)

(c) [2 marks] In the box below, give a brief discussion about the advantages / disadvantages between iteration and recursion.

Student ID: ....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

continued...

Student ID: ....

Question 7. Packing/Unpacking

[10 marks]

See the following code. Structure variable `astudent` holds the information about a student, and `pstu` is a pointer to `astudent`.

```
typedef struct student {
  int id;
  int age;
  char gender;
} Student;

Student astudent = {12345678, 20, 'F'};
Student *pstu = &astudent;
```

(a) [5 marks] Write a function `pack`, which packs all the data members in `astudent` into an `int` variable. The function prototype is given as follows:

```
int pack(Student *);
```

The function takes a pointer to `astudent` and returns an integer which contains the packed data. In this integer, you must use **1 bit** to store `astudent.gender`, **7 bits** for `astudent.age` and **24 bits** for `astudent.id`.

(Question 7 continued on next page)

(Question 7 continued)

(b) [5 marks] Write a function `unpack`, which unpacks the integer previously returned by `pack` into a `Student` variable. The function prototype is given as follows:

```
Student unpack(int);
```

The function takes the integer and returns the `Student` varaible which contains the unpacked data. After the unpacking, the values in the `Student` varaible must exactly match the values in `astudent`. You may use `pow()` to help create the masks that you need. The function prototype of `pow` is given in Appendix B.

continued...

Student ID: ....

Question 8. File Handling

[10 marks]

Suppose you have created a singly-linked list. Each of the nodes is a structure created by using the following type definition:

```
typedef struct node
{ char data;
  struct node *next;
} Node;
```

Also suppose you have a pointer to `Node`, `head`, which points to the first node in the list.

```
Node *head; /* head points to the first node in the list */
```

(Question 8 continued on next page)

Student ID: ....

(Question 8 continued)

(a) [5 marks] Write a function `writelistintext` which uses `fprintf()` to write each of the nodes in plain text to the file `list.dat`. For example, suppose in node 1 the value in `data` is `t` (character) and the value in `next` is `bb902068` (hexadecimal), and in node 2 it is `h` and `bb902070`, respectively. In the file (`list.dat`), it should look like this:

```
t        bb902068
h        bb902070
...
```

The function prototype is given as follows:

```
void writelistintext(Node *);
```

The function takes `head` (a pointer to `Node`) as the actual argument. You need to include an error message if the file cannot be opened. The function prototype of `fprintf` is given in Appendix B.

(Question 8 continued on next page)

(Question 8 continued)

(b) [5 marks] Write a function `writelisttofile` which uses `fwrite()` to write each of the nodes as a block of data to the file `list.dat`. The function prototype is given as follows:

```
void writelisttofile(Node *);
```

The function takes `head` (a pointer to `Node`) as the actual argument. You need to include an error message if the file cannot be opened. The function prototype of `fwrite` is given in Appendix B.

Student ID: ....

SPARE PAGE FOR EXTRA ANSWERS

Cross out rough working that you do not want marked.
Specify the question number for work that you do want marked.

continued...

Student ID: ....

A Operator Precedence

1 `(), ->, [], .`

2 `~, ++, --, + (unary), - (unary), *(unary), &(unary) , sizeof, (type)`

3 `*, /, % (arithmetic binary`

4 `+, - (arithmetic binary)`

5 `<<, >>`

6 `<, <=, >, >=`

7 `==, !=`

8 `&`

9 `^`

10 |

11 `&&`

12 ||

13 `?:`

14 `=, +=, -=, ... (assignment)`

15 `,`

B Useful Function Prototypes

1 `int fprintf(FILE *fp, const char *cntrl_string, ...);`
`fprintf()` writes formatted text into the file associated with `fp` and returns the number of characters written.

2 `unsigned fwrite(const void *ptr, unsigned size, unsigned nmemb, FILE *stream);`
`fwrite()` writes `nmemb` objects, each `size` bytes long, to the stream pointed to by `stream`, obtaining them from the location given by `ptr`.

3 `double pow(double x, double y);`
`pow()` computes the value of x to the exponent y.

C A Copy of sh Script for Question 1(b)

```
#!/bin/sh
#
# [Comment 1]
#
PROG='basename $0'
#
# [Comment 2]
#
SMSGATEWAY='smsuser@gateway.somehost.co.nz'
#
# [Comment 3]
#
```

```
DEST=$1
TEXT=$2
#
# [Comment 4]
#
SENDER='grep $USER /etc/passwd | cut -d: -f 5| sed -e 's/,.*//''
#
# [Comment 5]
#
if [ -z "$DEST" ]
then
    printf "Destination: "
    read DEST
fi

if [ -z "$TEXT" ]; then
   printf "Text: "
   read TEXT
fi

#
# [Comment 6]
#
MESSAGE_SIZE='echo "From $SENDER: $TEXT" | wc -c'

#
# [Comment 7]
#
if [ $MESSAGE_SIZE -le 160 ]
then
    #
    # [Comment 8]
    #
    ssh $SMSGATEWAY "sendsms $DEST \"From $SENDER: $TEXT\"" > /dev/null 2>&1
    #
    # [Comment 9]
    #
    if [ $? -ne 0 ]; then
        echo "$PROG: SMS not sent - could not contact gateway"
    fi
else
    #
    # [Comment 10]
    #
    echo "$PROG: SMS not sent - message is > 160"
fi
```

D Useful Unix Commands

1 BASENAME(1)
NAME: basename, dirname – return filename or directory portion of pathname
SYNOPSIS: basename string [suffix], dirname string

2 CUT(1)
NAME: cut – select portions of each line of a file

SYNOPSIS: cut -b list [-n] [file ...], cut -c list [file ...], cut -f list [-d delim] [-s] [file ...]

3 GREP(1)

NAME: grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS: grep [options] PATTERN [FILE...], grep [options] [-e PATTERN | -f FILE] [FILE...]

4 SED(1)

NAME: sed – stream editor

SYNOPSIS: sed [-aEnr] command [file ...], sed [-aEnr] [-e command] [-f command_file] [file ...]

5 SSH(1)

NAME: ssh – OpenSSH SSH client (remote login program)

SYNOPSIS: ssh [-1246AaCfgKkMNnqsTtVvXxY] [-b bind_address] [-c cipher_spec] [-D [bind_address:]port] [-e escape_char] [-F configfile] [-i identity_file] [-L [bind_address:]port:host:hostport] [-l login_name] [-m mac_spec] [-O ctl_cmd] [-o option] [-p port] [-R [bind_address:]port:host:hostport] [-S ctl_path] [-w local_tun[:remote_tun]] [user@]hostname [command]

6 WC(1)

NAME: wc – word, line, and byte count

SYNOPSIS: wc [-c | -m] [-lw] [file ...]

****