TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI

VICTORIA
UNIVERSITY OF WELLINGTON

EXAMINATIONS — 2014
TRIMESTER ONE

NWEN 241

SYSTEMS PROGRAMMING

Time allowed: THREE HOURS

Instructions: Closed Book

The examination contains 5 questions. You must answer ALL questions

The exam consists of 150 marks in total, distributed across each of the questions as follows:

Question 1 C General Questions	[40 marks]
Question 2 Arrays and Pointers	[40 marks]
Question 3 Data Structures	[24 marks]
Question 4 Bitwise Operators	[22 marks]
Question 5 File Handling	[24 marks]

No calculators are allowed.

No electronic dictionaries are allowed.

Paper foreign to English language dictionaries are allowed.

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.

Question 1 C General Questions **[40 marks]**

(a) [5 Marks] Explain the four steps of compilation for C programs.

(b) [5 Marks] C provides static and automatic storage classes. Explain how a variable can be declared to be static or automatic, and explain the difference in behaviour between the two classes.

(c) [5 Marks] Discuss the advantages and disadvantages of iteration versus recursion in C.

(d) [5 Marks] Assume the following malloc is successful:

```
int *ptr = malloc(20 * sizeof(int)); /* successful request */
```

Describe the possible outcomes of the following statement and discuss why a temporary variable `tmp` is used:

```
int *tmp = realloc(ptr, 200 * sizeof(int));
```

(e) [5 Marks] State what the problems the following program may have, with clear justifications.

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(void) {
  int i;
  char f1[] = "pear", f2[] = "apple", f3[] = "orange";
  char v1[6] = "tomato";

  strcpy(f1, f3);
  strcat(f1, f2);

  for(i=0 ; i<SIZE; i++) {
    printf("%d=%c\n", i, v1[i]);
  }

  return 0;
}
```

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.

(f) [4 Marks] Consider the following declaration.

```
int r1, r2, (*fp1)(void), (*fp2)(void), func(void);
```

State whether each of the four statements is valid or invalid.

```
fp1 = func;
```

```
fp2 = &func;
```

```
r1 = (*fp1)(void);
```

```
r2 = fp2(void);
```

(g) [4 Marks] Consider the following declarations.

```
int func(int **);

int a[2][4], (*b)[10], *c[20], **d;
```

Which of `a`, `b`, `c` or `d` could be passed as an argument to `func`? Explain your answer.

(h) [4 Marks] Suppose you are working on a 32-bit machine, where `sizeof(int)` is FOUR bytes, `sizeof(char)` is ONE byte, and `sizeof(int *)` is FOUR bytes.

Consider the following code.

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

mPchar ma, mb;
tPchar ta, tb;
```

Give the outputs of the following `printf` statements.

```
printf("%d ", sizeof(ma));
```

```
printf("%d ", sizeof(ta));
```

```
printf("%d ", sizeof(mb));
```

```
printf("%d ", sizeof(tb));
```

(i) [3 Marks] Using the same machine as in (h), consider the following code.

```
typedef struct {
int age;
char gender;
char *name;
} Person;

typedef union {
int i;
char c;
char *p;
} int_char;
```

Give the outputs of the following `printf` statements.

```
printf("%d ", sizeof(Person));
```

```
printf("%d ", sizeof(int_char));
```

Question 2 Arrays and Pointers **[40 marks]**

(a) [5 Marks] Implement function `func`, with prototype `void func(char *)`, which will print out a sequence of suffixes of its argument in decreasing size. For example, if `func` was passed a string containing `"Monday"`, it would print out:

```
Monday, onday, nday, day, ay, y,
```

(b) [14 Marks] Consider the following code.

```
char m[5][7] = {"abcdef", "ghijkl", "mnopqr", "stuvwx", "yz"};
char (*p)[7] = m;
```

Give the outputs of the following `printf` statements.

```
printf("%c", **m);
```

```
printf("%c", *(*m+2));
```

```
printf("%c", *(*(m+1)+1));
```

```
printf("%c", *(m[1]+2));
```

```
printf("%c", (*(m+2))[3]);
```

```
printf("%c", (*(p+3))[2]);
```

```
p++; printf("%d", (int)p - (int)m);
```

(c) [10 Marks] Give a declaration for the variable `p` in each of the following cases.

`p` is a pointer to a `char`.

`p` is a constant pointer to a `char`.

`p` is an array of 5 pointers to `char`.

`p` is pointer to function that takes no arguments and returns an `int`.

`p` is a pointer to an array of 10 `int` elements.

`p` is an array of 5 pointers to a function that takes no arguments and returns a pointer to an `int`.

`p` is a pointer to an array of 5 pointers to a function that takes no arguments and returns a pointer to a function that takes an `int` argument and returns an `int`.

(d) [5 Marks] Write a definition of the function `func` that takes a character and a string as its two arguments. `func` compares the character with the string. The function should return a nonzero value if the character is in the string and zero otherwise. You should take into account that the string should not be accidentally modified by the operations of the function.

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.

(e) [6 Marks] Consider the following code.

```
int main(void)
{
  char a1[] = "black", a2[] = "white";

  swap(a1, a2);

  printf("%s, %s \n", a1, a2);

  return 0;
}
```

Implement function `swap`, which swaps the values of two strings. The outputs of the above program should look like this:

```
white, black
```

You may assume the two strings are always the same size.

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.

Question 3 Data Structures **[24 marks]**

(a) [8 Marks] A person's height can be expressed in either metric or Imperial units. In metric units we use metres and we give a floating point number, e.g. 1.75 m. In Imperial units we use feet and inches and we give two integers, e.g. 5 feet and 8 inches. The following C type definitions allow us to represent a height:

```
typedef enum {metric, imperial} Scale;

typedef struct {
  int feet;
  int inches;
} FeetAndInches;

typedef union{
  float metres;
  FeetAndInches feetandinches;
} Value;

typedef struct {
  Scale scale;
  Value reading;
} Height;
```

It is useful to be able to convert a height from feet and inches to metres. Define a function with prototype `void tometres(Height *h)`; Suppose that `h` is a height, which may be in either metres or feet and inches. After `tometres(&h)` has been called, `h` should be expressed in metres. One foot is 0.3048 metres, and one inch is exactly 0.0254 metres.

(b) The following type definitions, macro definition and function prototypes are part of a queue model, where we are using singly-linked lists to implement queues.

```
#define Node_Size sizeof(Node)

typedef char Data;

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

typedef Node *ptrNode;

typedef struct queue
{ int cnt;        /* counts the number of nodes */
  ptrNode front;  /* points to the front node */
  ptrNode rear;   /* points to the rear node */
} Queue;

void enqueue(Data, Queue *);
Data dequeue(Queue *);
```

Suppose the queue that we have implemented has a header node of type `Queue` and a list of linked nodes of type `Node`. The pointer `front` in the header node points to the front node in the list, while the pointer `rear` points to the rear node in the list. The header node also has a counter `cnt`, which counts the number of nodes in the list.

i. [8 Marks] Write C code to implement the function `enqueue`, which adds a new node to the rear of the list. The character passed to `enqueue` needs to be assigned to the variable `data` in the node. You can assume your requests for memory are always successful. You DO NOT need to consider the case when the queue is empty or full.

ii. [8 Marks] Write C code to implement the function `dequeue`, which deletes the front node from the list. The character stored in data needs to be returned. You DO NOT need to consider the case when the queue is empty, and you may assume the queue is never full.

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.

Question 4 Bitwise Operators **[22 marks]**

(a) [8 Marks] In the following, we have defined a structure type named Student:

```
typedef struct student {
  int id;
  int age;
  char gender;    /* Either 'M' or 'F' */
} Student;
```

Define a function with prototype `int pack(Student *);` which packs all the data members in a `Student` variable into an `int` variable and returns the value of this `int` variable. In this `int` variable, you must use **1 bit** to store `gender`, **7 bits** for `age` and **24 bits** for `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.

(b) [6 Marks] Write a definition of the function `bitwise_swap` that uses only bitwise operators to swap the values of two integers.

(c) [8 Marks] Suppose you are working on a 32-bit machine. Write a program that prints each bit of an integer. The program should get the user to type in an integer, and then should print the integer in the following format with spaces between blocks of 8 bits and a new line character at the end.

```
01001000 01101101 00001111 00010111
```

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.

Question 5 File Handling **[24 marks]**

(a) [8 Marks] Write a command-line-arguments based program. The program will be called with two file names as its command line arguments. The program should read the text from the first file and write it to the second file.

You need to implement the program with the `main()` function.

(b) For this question you need to write two functions which will write and read a singly-linked list to / from a binary file. The singly-linked list is constructed of nodes of the following `Node` type.

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

Assume the function prototypes of `fwrite` and `fread` are as follows:

```
int fwrite(void *, int, int, FILE *);
int fread(void *, int, int, FILE *);
```

i. [8 Marks] Define a function with prototype `void writelisttofile(Node *);` which uses `fwrite()` to write each of the nodes as a block of data to the file `list.dat`. You need include an error message if the file cannot be opened.

ii. [8 Marks] Define a function with prototype `void readlistfromfile(void);` which uses `fread()` to read each of the nodes (a block of data) from the file `list.dat` and prints them on screen. For example, suppose the `data` value of the first node was `t` (character) and `next` was `bb902068` (hexadecimal), and the second node had `h` and `bb902070`. The output should look like this:

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

You may assume that the file can always be successfully opened.

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.

* * * * * * * * * * * * * * * * * * *