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

EXAMINATIONS — 2006

MIDTERM TEST

COMP 206

PROGRAM AND DATA STRUCTURES

Time Allowed: 90 minutes

Instructions: You should attempt all the questions.
The exam will be marked out of 90. **=87+3!**
Write your answers in the exam paper and hand in all sheets.
Operator precedences are given in Appendix A.

Student ID: .....

Question 1. [24 marks]

(a) [8 marks] Suppose that `a` has been declared:

```
char a[] = "COMP206";
```

State the value of each of the following expressions:

```
a => "COMP206"
```

2 marks

```
*a => C (or 67)
```

2 marks

```
*(a+7) => null character (or '\0')
```

2 marks

```
*a+7 => J (or 74)
```

2 marks

(Question 1 continued on next page)

Student ID: .....

(Question 1 continued)

(b) [6 marks] I wrote the following code to read a name and an age from the terminal and to write them back out:

```
#include <stdio.h>

int main()
{
  int age;
  char name[20];

  printf("Enter the person's name and his or her age:\n");
  scanf("%s %d", name, age);
  printf("\nName = %s\nAge= %d\n", name, age);

  return 0;
}
```

The code compiles, but causes a runtime error when executed:

```
circa % gcc -o qn1 qn1.c
circa % ./qn1
Enter the person's name and his or her age:
Noah 950
Segmentation fault
```

In the box below, explain what the problem is, and how to fix my code.

C uses call-by-value, so the actual parameters of a function cannot be changed by the function call. If we want to use a function to update the value stored in a variable we need to pass in a reference (i.e. a pointer) to the variable. The purpose of `scanf` is to read in values so `scanf` needs to be supplied with pointers: `name` is OK, but we need `&age`:

```
scanf("%s %d", name, &age);
```

Marks: 4 for the explanation, 2 for the code.

(Question 1 continued on next page)

Student ID: .....

(Question 1 continued)

(c) [10 marks] You want to write a C function `swap` which swaps the values of two integer variables. Fill in the boxes in the following program:

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

```
/* swap prototype */
void swap(int *px, int *py);
```

2 marks

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

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

```
/* call to swap x and y */
swap(&x, &y);
```

2 marks

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

  return 0;
}
```

Student ID: .....

```
/* swap definition */
void swap(int *px, int *py){
  int tmp;

   tmp=*px;
   *px=*py;
   *py=tmp;
   return;
}
```

6 marks

This output should be generated:

```
x = 0
y = 1
x = 1
y = 0
```

Student ID: .....

Question 2. [14 marks]

A person's weight can be expressed in either kilograms (kg) or pounds (lbs).

- If we express a weight in kilograms we use a floating point number, e.g. 75.5 kg.
- If we express a weight in pounds we use an integer, e.g. 165 lbs.

Th following C type definitions allow us to represent a weight:

```
typedef enum {kilos, pounds} Scale;

typedef union{
    float kg;
    int lbs;
} Value;

typedef struct {
    Scale unit;
    Value reading;
} Weight;
```

(a) [6 marks] Define a function with prototype `void printweight(Weight w);` which will print out a weight, stating whether it is a weight in kilograms or pounds.

```
void printweight (Weight w){
    switch (w.sort){
            case pounds: printf("%d pounds\n", w.reading.lbs);
                         break;
            case kilos: printf("%f kgs\n", w.reading.kg);
                         break;
    }
}
```

6 marks
Using `if(w.sort == pounds){...}` was also OK.

(Question 2 continued on next page)

Student ID: .....

(Question 2 continued)

(b) [8 marks] It is useful to be able to convert a weight from kilograms to pounds. Define a function with prototype `void topounds(Weight *w);`

Suppose that `v` is a weight, which may be in either kilograms or pounds. After `topounds(&v)` has been called `v` should be expressed in pounds. There are 2.2 pounds in a kilogram.

```
void topounds(Weight *w){
    switch (w->sort){
           case pounds: break;
           case kilos: { w->sort=pounds;
                         w->reading.lbs=w->reading.kg*2.2;
                         break;
                         }
    }
}
```

Using `(*w).sort` etc. instead of `w->sort` was OK, as was using `if` rather than `switch`.

Student ID: .....

Question 3. [18 marks]

(a) [6 marks] You have been hired as a C programmer to lead a programming team. Unfortunately, although your colleagues are experienced Java programmers, they know nothing about C, and you soon realise that you must help them to understand C.

In the box below state what you consider to be the three most important differences between C and Java

2 marks each for sensible answers, such as:

- Java supports object oriented programming, whereas C does not;
- Java compiles to bytecodes which are executed on the Java Virtual Machine, thereby providing a degree of machine independence. C lacks this degree of machine independence as C programs are compiled to instructions for a specific architecture;
- C allows/forces the programmer to manage dynamically allocated memory.

It was up to the markers to decide if an answer was sensible or not.

(b) [6 marks]

In the box below give the prototype for `malloc` and a description of its behaviour.

From the BSD Library Functions Manual:

```
void *malloc(size_t size);
```

The `malloc()` function allocates **size** bytes of memory and returns a pointer to the allocated memory. `malloc()` returns a `NULL` pointer if there is an error.

2 marks for the prototype. Allocating size bytes, returning a pointer, and what happens in the event of failure to allocate were what got the other marks.

(Question 3 continued on next page)

(Question 3 continued)

(c) [6 marks] Explain how a make file helps the programmer when writing large programs.

Basically provide some support for abstraction (and hence automation) in the compilation process. 2 marks each for points such as:

- provides automated support – complex collections of commands reduced to one command
- since make files allow the programmer to express dependencies between files code can be structured to exploit this support
- allows for easy customisation – variables can be used to specify e.g. architecture, location of compiler, location of libraries, compiler flags
- keep track of dependencies, and only re-compile what needs to be recompiled
- provides documentation for programmer

Student ID: .....

Question 4. [21 marks]

The following C program defines and operates on a datatype of lists of integers. Fill in the boxes to complete the program. It should output `Sum = 55`.

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

#define list_struct_size sizeof(struct list)


typedef enum {false=0, true=1} bool;

typedef struct list {
    int value;
    struct list *next;
} *ListPtr;


int suml(ListPtr pl);              /* Add all the items in a list */
void enlist(ListPtr* ppl, int i); /* Put an item into a list */
bool isempty(const ListPtr pl);   /* Test if a list is the empty list */


int main()
{
    ListPtr lp = NULL;
    int i;

    for(i=1; i <= 10; enlist(&lp, i), i++);

    printf("Sum = %d\n", suml(lp));

    return 0;
}
```

(Question 4 continued on next page)

Student ID: .....

(Question 4 continued)

```
int suml(ListPtr pl){
    int sum;
    sum = 0;

    while(pl!=NULL){
        sum+=pl->value;
        pl=pl->next;
    }
    return sum;
}
```

8 marks

```
void enlist(ListPtr* ppl, int i) {
    ListPtr tmp;

    tmp = malloc(list_struct_size);
    if(tmp==NULL){exit -1;}

    tmp->value = i;
    tmp->next = *ppl;
    *ppl = tmp;
}
```

8 marks

(Question 4 continued on next page)

Student ID: .....

(Question 4 continued)

```
bool isempty(const ListPtr pl){
    return(pl==NULL);
}
```

5 marks

Student ID: .....

Question 5. C++ Fundamentals [13 marks]

(a) [10 marks] Prior to decimalisation on 10 July 1967, New Zealand used a system of currency made up of pounds, shillings and pence. One pound was equivalent to 20 shillings, and 12 pence made up one shilling.

The class definition below describes values in this currency.

```
class Value {
 public:
  Value();
  ~Value();
  void setValue (int l, int s, int d);
  void printValue();
 private:
  int pounds;
  int shillings;
  int pence;
};
```

Identify the constructor, destructor, member functions, data members, and the member access specifiers:

`Constructor: Value`

2 marks

`Destructor: ~Value`

2 marks

`Member functions: void setValue (int l, int s, int d), void printValue()`

2 marks

(Question 5 continued on next page)

Student ID: .....

(Question 5 continued)

Data members:int pounds, int shillings, int pence

2 marks

Member access specifiers:private, public

2 marks

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 ,
```

****