Curtin University — Department of Computing

UNIX and C Programming (COMP1000)

Mock Test 1A

Practice for Test 1

Real test weight: 15% of the unit mark.

Attempt this mock test in preparation for Test 1. Answer all questions by yourself.

Will the real test be like this?

The real test will follow approximately the same format and will cover the same material. However, the questions will be different (so trying to memorise answers will get you nowhere!) The real test will also be closed book – no books, notes, electronic devices, etc.

How can I get help/feedback?

First, make your best attempt. Then, to obtain feedback, see your tutor, or the senior tutor, or the lecturer.

Will you upload the answers?

No. Sample answers to this mock test *will not* be provided – no exceptions.

Why?

Sample answers *discourage* people from putting in real effort to learn the concepts and skills. They encourage rote (fake) learning, where you try to memorise an answer without understanding how to obtain it or even why it's correct.

Basically, if you're given the answers, it's too easy to convince yourself that you don't need to work them out.

Updated: 28th August, 2015

Question 1 (8 marks)

Explain the meaning of the following:

(a)
```
void f() {
    static int x = 5;
    ...
}
```

(b)
```
int i = 22;
int* j = &i;
```

(c)
```
float *f(float** x, void* y);
```

(d)
```
void *(*p)(void*);
```

(e)
```
#define SEQ(a,b,c) ((a) < (b) && (b) < (c))
```

(f)
```
typedef void (*T)(double, int);
```

(g)
```
#ifdef C
    int* x;
#else
    double* x;
#endif
```

Question 2 appears on the next page

Question 2 (14 marks)

Say you have the following declarations:

```
float e = 2.0;
float f = 0.5;
float* g = NULL;
float* h = NULL;
float* i = NULL;
float** v = NULL;
float** w = NULL;
```

Based on each of the following code snippets:

- Draw a diagram showing all pointer relationships created.
- State the resulting values of **e** and **f**.

(a)
```
h = &e;
v = &h;
w = v;
i = *w;
v = &g;
g = &f;
*v = h;
**w = **v * **v;
```

(b)
```
int j;
v = &g;
w = &h;
*v = &e;
*w = &f;
i = *v;
for(j = 0; j < **w; j++) {
    *i = -*i;
}
**w = -**v;
```

Question 3 appears on the next page

Question 3 (8 marks)

Each of the following C functions contains an error. Explain what is wrong.
(You don't need to show how to fix it, but you can if it will assist your explanation.)

(a)
```c
static double counter(double* increment)
{
    /* Maintains a running counter, which is incremented
    and returned whenever the function is called. */
    double counter = 0;
    counter = counter + *increment;
    return counter;
}
```

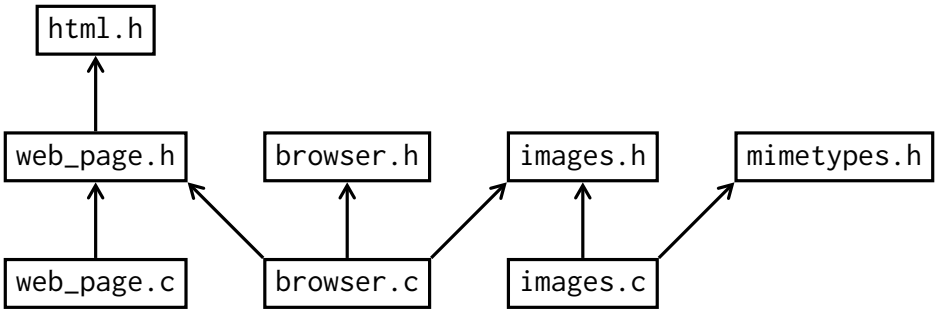
(b)
```c
int square(int n)
{
    int* x;
    *x = n * n;
    return *x;
}
```

(c)
```c
void readVals(int x, int y)
{
    printf("Enter first integer: ");
    scanf("%d", &x);
    printf("Enter second integer: ");
    scanf("%d", &y);
}
```

Question 4 appears on the next page

Question 4 (10 marks)

The following diagram represents the `#include` relationships between the files making up a simple web browser.



Produce a suitable makefile for this project. Make good use of makefile variables, and include a suitable "clean" rule. The executable file should be named `browser`.

Question 5 appears on the next page

Question 5 (40 marks)

NASA is designing a new space probe to search for life on other planets. The probe will be sent to a given planet. It will orbit the planet, map the surface and take sensor readings. The probe will look for promising locations on the surface for further exploration.

Your job is to design the software to analyse the probe's measurements. The probe will divide the planetary surface into a rectangular grid (excluding the polar regions). The non-negative grid coordinates x and y will identify each grid square. The probe will take one set of readings in each grid square. Specifically, it will record:

- temperature (°C),
- wind speed (km/h),
- oxygen concentration (%), and
- volcanic activity (yes/no).

For each set of readings (i.e. for each grid square), your software must compute a "life rating". For the planet/grid as a whole, it must also compute an "exploration risk" (an indicator of the danger faced by further, surface exploration).

If there is *no* volcanic activity in a given grid square, the life rating is as follows:

$$\text{life rating} = \frac{\text{temperature} \times \text{oxygen concentration}}{\text{wind speed}}$$

With volcanic activity, life rating is calculated the same way, but then divided by 3.5.

For each grid square containing volcanic activity, exploration risk goes up by 1.

Write a C function (not a whole program) to do the following:

1. Retrieve the sensor readings from the probe, using the **`getReadings()`** function.
2. Calculate the life rating and exploration risk, as described above.
3. Report each life rating by calling another function (through a given function pointer).
4. Report the exploration risk (via an `int*` parameter).

The **`getReadings()`** function is declared as follows:

```
void getReadings(int x, int y, double* temp, double* windSpeed,
                 double* oxygen, int* volcanism);
```

Your function should return `void` and take four parameters:

- **`width`** — an `int`, the width of the grid ($0 \leq$ `x` $<$ `width`).
- **`height`** — an `int`, the height of the grid ($0 \leq$ `y` $<$ `height`).
- **`explorationRisk`** — a pointer to an `int`.
- **`reportFunc`** — a pointer to a function used to report the life rating. The function takes two `int`s (grid coordinates) and a `double` (the life rating), and returns `void`.

End of Mock Test 1A