CSC 230 Section 001

Midterm test

February 14, 2007

Questions Q1 and Q4 carry 15 marks each, all other questions carry 10 marks each. This adds up to 80, but 75 is a perfect score. If you obtain more than 75, your marks will be added up to 75 and will be considered 100%.

It should be possible to answer the questions in the space provided. Use the back of the paper where you are specifically asked to. Otherwise, do not use the back of the paper or attach extra sheets unless you absolutely have to. You may consult any books, notes, etc., but no other person. You must not use a cellphone, computer, PDA, or any other information processing device.

This is a one hour test.

First Name: _____________________________
Last Name: _____________________________
Student ID: _____________________________

Pledge of Honor: I pledge on my honor that I have answered the questions in this test purely on my own individual ability, and have neither given help to nor received help from another person.

Signature: ________________________________

Answers

Grading Remarks
Q1. Consider the following complete program.

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

int i = 42;

int main(void)
{
    int i = 1;
    int j = 10;

    do {
        /* some code here, not changing i or j */
    } while (--j > i);

    i++;
    {
        i += j;
    }
    i = i + 2 * j - i * j;

    if (i*j > j) { /* let’s call this Block A */
        int i;
        i = 9;
        i = i + 2 * j - i * j;
    }

    printf("%d\n", i);
    return 0;
}
```

(a) Which variable’s value is being printed out by the final printf() statement? Answer by identifying the scope of that variable, and the name of the variable.

Ans: It is the variable called i, defined at the very beginning of the main() function, and has scope local to the entire function main().

(b) What value will be printed out by the program when it is run?

Ans: 2

Total 9. If name or scope is missing, deduct 3 in each case. If scope is incomplete, e.g. simpl “local” without specifying block, deduct 1 or 2 in each case. If scope is incorrect, e.g. i instead of j, deduct 2 in each case.

Q2. Fill in the blanks in the following statements, using the numbers corresponding to the options in brackets at the end, and writing in your own text if there are fewer options than blanks.

(a) Typically header (.h) files contain __________, and source (.c) files contain __________.
   1. “function definitions” 2. “function declarations”

(b) Three distinct stages of building an executable program are __________, __________, and __________.

(c) If the return value of a function is not specified by the source code defining or declaring that function, the compiler assumes that the function returns a __________.

(d) The preprocessor directive __________ causes the header file named after the directive to be bodily included in the source code before the result is passed to the compiler.

   (e) The process by which the compiler groups successive keywords, literals, identifiers, punctuators in preparation to parsing the source file is called __________.

Total 6 marks. 6 if correct, 0 if incorrect.
Q3. The following statement likely contains an error, in that it is likely to produce a result different from that expected. Explain what the error is and why it happens.

```
printf ("The number of students is %d.\n", 118.0);
```

“118.0” is a literal which indicates a real number, because of the inclusion of the decimal point, therefore it is a literal float, not int. When printf tries to interpret the bytes passed as representing an int variable, this is a misinterpretation and results in the wrong value being printed.

Any reasonable answer showing an appreciation of this issue is okay. If variable type mismatch is mentioned, give part credit. If the issue with printf not knowing what the type is other than from the format string is also mentioned, give full credit.

If the answer is that “it will truncate it to an integer”, then part mark, in recognition of the fact that the type mismatch has been acknowledged.

Irrelevant verbiage about what printf does or where it comes from should not earn much credit.
Q4. Given the declarations

```c
int i = 4;
int j = 6;
float f1 = 3.14159;
```

find the values of each of the following expressions if it is evaluated immediately after the declarations. You must also indicate the type of each evaluated value (int, float, etc.).

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>j + f1</td>
<td>9.14159</td>
<td>float</td>
</tr>
<tr>
<td>i + (j + f1)</td>
<td>13.14159</td>
<td>float</td>
</tr>
<tr>
<td>(i + (int)(j + f1))</td>
<td>13</td>
<td>int</td>
</tr>
<tr>
<td>(i + (int) j + f1)</td>
<td>13.14159</td>
<td>float</td>
</tr>
<tr>
<td>(int) (i + j + f1)</td>
<td>13</td>
<td>int</td>
</tr>
</tbody>
</table>

Q5. A source file called `new_test.c` uses functions whose definitions are available in another source file called `log_functions.c`, and also uses functions that are available in the math library. It is desired to compile this file to an executable called `new_test`. Write a single command line function, using the compiler `gcc`, to perform this action.

**Ans:**

```bash
gcc -lm new_test.c log_functions.c -o new_test
```

If any of the components of the command are left out, deduct points. If other options such as `--pedantic` are also used, this is fine. If `--c` is used, this is not okay – deduct points.

Q6. Write a statement that will cause the program to call a function `abort()` if the current value of two `int` variables `i` and `j` is such that multiplying these values together will result in a number greater than `INT_MAX`, the macro which indicates the maximum value an `int` variable can hold.

**Ans:**

```c
if ( (double) INT_MAX / (double) i < j)
    abort();
```

Other equivalent statements will do: however, note that

```c
if ( (long)i * (long) j > INT_MAX)
    abort();
```

- does not work. Because `long` may be no larger than `int`. This solution should earn part, but not all, marks, say 7. If assumption that `LONG_MAX` is greater than `INT_MAX` is explicitly stated, then full marks.

Failing to cast as double and just doing

```c
if (INT_MAX/j < i)
    abort();
```

- does not work, because this becomes integer division with the fractional part truncated, so fails to check the original condition. This solution should earn part marks, say 4.

Actually, this works just fine; it was a mistake on my part to say that this was wrong! Because `INT_MAX` is itself an `int`, and integer division always truncates, it will be an equivalent condition.
Q7. Re-write the following segment of code using a `do-while` loop instead of a `for` loop, and without using any side-effects. You are not allowed to define any new variables. `i` and `j` are integer variables, and `f` is a float variable. All of these variables should have the same values when your loop terminates as at the termination of the original one. You are allowed to write at most one statement outside your loop body, either before the beginning or after the end.

```c
for (j=i; (f=j/(float)i) > 0.5;) {
    printf (“Now trying %d
”, --j);
}
```

If new variables are used, deduct marks. There are several ways to go wrong in this one. Correct variations are okay, but there are not many simple yet correct variations. If the answer differs from this one, make sure it is correct (by using a small initial value for `i`, say 5) that it in fact does the right thing.

The three lines marked by comments are logically required, but in this particular case, since `i` and `j` start out being the same, the loop will at least run once, so it is okay to eliminate those lines; no penalty.

```c
j = i;
do {
    f = j / (float)i;          /*                               */
    if (f > 0.5)               /*  Strictly not needed, since   */
        break;               /*   j equals i at beginning     */
    --j;                      /*                               */
    printf (“Now trying %d
”, j);
    f = j/(float)i;
} while (f > 0.5);
```