

EECS 110 Midterm February 13, 2008

Don't panic! Read each question through. If any part confuses you, ask me privately. Watch your time. Don't spend too long on any one question. Write cleanly. If you need to make big changes, X out your answer, write "see back" and write your new version on the back, with the number of the question.

1. (6 pts) Show the output of the following program fragment:

Program	Output
<pre>for (int i = 1; i <= 4; ++i) { for (int j = 0; j <= 8; j += i) { printf("%d ", j); } printf("\n"); }</pre>	<pre>0 1 2 3 4 5 6 7 8 0 2 4 6 8 0 3 6 0 4 8</pre> <p><i>Common mistakes: no starting from 0, going past 8, not incrementing by 1, 2, 3, and 4</i></p>

2. (9 pts) Show the output of the following program fragment:

Program	Output
<pre>Show the output of the function call ack(1, 2) where ack() is defined thus: int ack(int m, int n) { printf("%d %d\n", m, n); if (m == 0) { return n + 1; } else if (m > 0 && n == 0) { return ack(m - 1, 1); } else { return ack(m - 1, ack(m, n - 1)); } }</pre>	<pre>1 2 1 1 1 0 0 1 0 2 0 3</pre> <p><i>Note: this is the Ackermann function. It is interesting to the theory of computation, because it starts to generate very large numbers very quickly. The Wikipedia entry is pretty good.</i></p>

3. (10 pts) The program below is supposed to calculate the average of a set of grades, ignoring the lowest grade, but it's full of mistakes. **Circle** every error you can find. **Write** the correct code next to it. **Write COM if and only** if the mistake will cause a compilation error. **Write RUN** if the mistake will compile and run but do the wrong thing. Assume the missing function `get_low_grade` correctly returns the index of the lowest grade.

```
#define MAX_GRADES 5

int main()
{
    int g[MAX_GRADES] = [ 65; 72; 81; 60; 91 ];
    int low = get_low_grade(g[MAX_GRADES], size);
    double average = average_no_low(g[MAX_GRADES], size, low);

    printf("Average = %d/n", average);

    return 0;
}

/* calculate average, skipping g[low] */
double average_no_low(int g, int n, int low) {

    int sum;

    for (int i = 1; i <= n; i + 1) {
        if (i < low) {
            sum + g[i];
        }
    }

    return sum / n - 1;
}
```

Comment [CKR1]: COM Insert
#include <stdio.h> (no points taken off)

Comment [CKR2]: COM Delete =

Comment [CKR3]: COM Insert
average_no_low prototype

Comment [CKR4]: COM; should be ,

Comment [CKR5]: COM [] should be {}

Comment [CKR6]: COM Delete
[MAX_GRADES]

Comment [CKR7]: COM: size needs to be declared with a value.

Comment [CKR8]: COM Delete
[MAX_GRADES]

Comment [CKR9]: Should be %f

Comment [CKR10]: RUN Should be \

Comment [CKR11]: COM should be g[]

Comment [CKR12]: RUN Insert = 0

Comment [CKR13]: RUN Should be 0

Comment [CKR14]: RUN Should be <

Comment [CKR15]: RUN Should be i++

Comment [CKR16]: RUN should be !=

Comment [CKR17]: RUN should be +=

Comment [CKR18]: RUN insert (double)

Comment [CKR19]: RUN needs ()

I gave 1 point for each real mistake found. Common errors:

- *Calling functions with g[] instead of g;*
- *Putting character quotes on numbers, e.g., { '65', '72', ... }*
- *Replacing n with size – the n was just fine*
- *Declaring size and using it but not giving it a value*
- *Labeling the errors in average_no_low as COM – most were RUN*

4. (7 pts) Define a function `min_of_3` that takes 3 integer parameters, `x`, `y`, and `z` and returns the smallest. It should not define any additional variables.

```
int min_of_3(int a, int b, int c)
{
    if (a < b && a < c) {
        return a;
    } // at this point, either b <= a or c <= a, or both
    else if (b < c) {
        return b;
    }
    else {
        return c;
    }
}
```

Common mistakes:

- *Not handling duplicates, e.g., 1, 1, 3 or 3, 1, 1.*
- *Printing instead of returning.*
- *Doubly-declaring the input parameters as local variables.*

5. (8 pts) Define a function `get_low_grade` that takes two parameters, an array of grades, `g`, and an integer `n` with the number of grades. `get_low_grade` should return the index of the lowest grade. `n` will always be at least 1. If a low grade repeats, the index of the first one should be returned. Example: for `[75, 91, 62, 83, 62, 88]`, `get_low_grade` should return 2.

```
int get_low_grade(int g[], int n)
{
    int low = 0;

    for (int i = 1; i < n; ++i) {
        if (g[i] < g[low]) {
            low = i;
        }
    }
    return low;
}
```

Comment [CKR20]: Common mistake

Common mistakes:

- *Not declaring loop variable.*
- *Using nested for loops but not saving results.*
- *Assuming some maximum value (allowed but unnecessary)*
- *Doubly-declaring the input parameters as local variables*