ECE 264 Exam 1

6:30-7:30PM, February 16, 2011

I certify that I will not receive nor provide aid to any other student for this exam.

Signature:

You must sign here. Otherwise you will receive a **1-point** *penalty.*

Please write legibly. Your exam is not graded if your writing is hard to read.

This exam is printed **double sided**. Please read the questions carefully. Two common mistakes are answering a wrong question and failing to answer all questions.

This is an *open-book, open-note* exam. You can use any book or note or program printouts.
Please turn off your cellular phone and iPod. No electronic device is allowed.

Contents

Total Score: out of 15.

This page is blank. You can write answers here.

1 C Control Flow (3 points)

1.1 `if`

```
int x = 2;
int y = 6;
int z = 4;
int a = 0;
if ((x > 0) && (y < 0)) {
    if (z != 4) {
        a = 10;
    }
 }
 else {
     if ((x < y) || (z == 4)) {
         a = 20;
     }
}
```

What is the value of `a` after executing the above code? (1 point)

1.2 Function Call

```
#include <stdio.h>
int f(int i, int j, int k)
{
  i = j + k;
  j = 5;
  return (j * k);
}

int main(int argc, char * argv[])
{
  int a = 2;
  int b = 6;
  int c = 4;
  int d = 0;
  d = f(a, b, c);
  printf("%d %d %d %d\n", a, b, c, d);
  return 0;
}
```

What are the values of `a`, `b`, `c`, and `d` after executing the above code? (1 point)

You do not have to calculate the final answer. Instead, you can write down how to calculate the answer. For example, you can write `1 + 2` instead of `3`.

1.3 Variable Scope

```
#include <stdio.h>
int main(int argc, char * argv[])
{
  int cnt = 0;
  int a = 2;
  int b = 4;
  for (cnt = 2; cnt < 10; cnt += 2)
    {
      int a = 0;
      a += cnt;
      b += a;
    }
  printf("%d %d\n", a, b);
  return 0;
}
```

What are the values of `a` and `b` after executing the above code? (1 point)

You do not have to calculate the final answer. Instead, you can write down how to calculate the answer. For example, you can write `1 + 2` instead of `3`.

2 Recursion (4 points)

Recursion is a powerful algorithm strategy for many types of problems where you define the solution in terms of a **single step** of the algorithm. If you are able to write the solution for the **base step**, i.e. for the starting point of the algorithm, and for the **recursive step**, i.e. for an arbitrary step in the algorithm, then the concept of *induction* states that your solution will work for **all** steps.

2.1 Factorial

For example, to calculate the factorial $n!$, we can use the following recursive code:

```
int nfac(int n)
{
    // Base step: n is zero
    if (n == 0) return 1;

    // Recursive step: n greater than zero
    return n * nfac(n - 1);
}
```

What is the value of `a` after calling the following code? (1 point)

```
int a;
a = nfac(-2);
```

2.2 Fibonacci numbers

The Fibonacci number F_n is defined as:

$$F_0 = 0 \quad (1)$$
$$F_1 = 1 \quad (2)$$
$$F_n = F_{n-1} + F_{n-2} \quad (3)$$

Write a recursive function `int fib(int n)` that calculates the Fibonacci number for n. (2 points)

2.3 Recursive Calls

```
#include <stdio.h>
void f(int n)
{
  int i;
  if (n <= 0) { return; }
  for (i = 1; i <= n; i ++)
    {
      f(n - i);
    }
}

int main(int argc, char * argv[])
{
  int i;
  f(4);
  return 0;
}
```

How many times `f(2)` (i.e., `n` is 2) is called? (1 point)

3 Vector (3 points)

Write a constructor, a function to add two vectors, and another function that returns the scalar product for the structure `Vector`.

```
/* vector.h */
#ifndef VECTOR_H
#define VECTOR_H
typedef struct
{
  double x;
  double y;
  double z;
} Vector;
Vector Vector_construct(double a, double b, double c);
/*
  return a Vector whose
  x value is a
  y value is b
  z value is c
*/

Vector Vector_add(Vector v1, Vector v2);
/*
  return a Vector object whose
  x value is the sum of v1's x and v2's x
  y value is the sum of v1's y and v2's y
  z value is the sum of v1's z and v2's z
*/

Vector Vector_scalar(Vector v, double s);
/*
  return a Vector object whose
  x value is the product of v's x and s
  y value is the product of v's y and s
  z value is the product of v's z and s
*/

#endif /* VECTOR_H */
/* end of vector.h */

/* ========================================== */
/* vector.c */

Vector Vector_construct(double a, double b, double c)
{
```

```
  /* ===> FILL IN CODE BELOW <=== */









}

Vector Vector_add(Vector v1, Vector v2)
{
  /* ===> FILL IN CODE BELOW <=== */









}

Vector Vector_scalar(Vector v, double s)
{
  /* ===> FILL IN CODE BELOW <=== */









}
/* end of vector.c */

/* ========================================== */
/* main.c */

int main(int argc, char *argv[])
{
  /* create four Vector objects */
  Vector v1, v2, v3, v4;
```

```
  /* v1's x is 0.2, y is 0.5, z is 0.8 */
  /* ===> FILL IN CODE BELOW <=== */
  v1 =




  /* v2's x is 0.1, y is -0.3, z is 1.9 */
  /* ===> FILL IN CODE BELOW <=== */
  v2 =




  /* v3 is the sum of v1 and v2, use the Vector_add function  */
  /* ===> FILL IN CODE BELOW <=== */
  v3 =




  /* v4 is the product of v1 and 2.5 */
  /* ===> FILL IN CODE BELOW <=== */
  v4 =




  /* change v1's x value to 26.4 */
  /* ===> FILL IN CODE BELOW <=== */




  return 0;
}
```

4 Pointers (5 points)

```
#include <stdio.h>
void f1(int arg1, int arg2, int arg3)
{
  arg1 = arg2 * arg3;
}

void f2(int * arg1, int * arg2, int * arg3)
{
  arg1 = arg2;
  * arg1 = * arg3;
}

void f3(int * arg1, int * arg2, int * arg3)
{
  int * temp;
  temp = arg3;
  arg3 = arg2;
  arg2 = temp;
}

void f4(int * arg1, int * arg2, int * arg3)
{
  * arg1 = (* arg2) * (* arg3);
}

void f5(int * arg1, int * arg2, int * arg3)
{
  arg1 = arg2;
}

int main(int argc, char * argv[])
{
  int a1 = 2;  int b1 = 6;  int c1 = 4;
  int * pa1 = & a1;
  int * pb1 = & b1;
  int * pc1 = & c1;
  int a2 = 2;  int b2 = 6;  int c2 = 4;
  int a3 = 2;  int b3 = 6;  int c3 = 4;
  int a4 = 2;  int b4 = 6;  int c4 = 4;
  f1 (a1, b1, c1);
  /* What is the value of a1, b1, and c1 here, after calling f1?
     (1 point) */
```

```
  f2 (pa1, pb1, pc1); /* === CAUTION pa1 === */
  /* What is the value of a1, b1, and c1 here? (1 point) */



  f3 (& a2, & b2, & c2);
  /* What is the value of a2, b2, and c2 here? (1 point) */



  f4 (& a3, & b3, & c3);
  /* What is the value of a3, b3, and c3 here? (1 point) */



  f5 (& a4, & b4, & c4);
  /* What is the value of a4, b4, and c4 here? (1 point) */



  return 0;
}
```