ECE 264-1 Exam 2

01:30-02:20PM, March 10, 2010

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 **1-point** *penalty.*

This exam is printed **double sides**. Please read the questions carefully. Two common mistakes are answering wrong questions 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.

Outcome 2 is evaluated in two questions. To pass the outcome, you need to obtain 50% or more of the total scores in the two questions.

Contents

Total Score: out of 20.

Passed Outcome 2: Yes No

This page is blank. You can write answers here.

1 Pointer (6 points)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct
{
  char * name;
  int age;
} Person;
int f(int * * a, int * b, int c)
{
  * a = b;
  * b = c;
  c = 5;
  return 3;
}
void g(Person * * s, Person * t, Person u)
{
  * s = t;
  t = & u;
  strcpy(u.name, "purdue");
}
int main(int argc, char * argv[])
{
  int w = 2;
  int x = 6;
  int * y = & x;
  int z = 8;
  w = f (& y, & x, z);
  printf("w = %d, x = %d, y = %d, z = %d\n", w, x, * y, z);

  Person p1;
  p1.name = malloc(sizeof(char) * 80);
  strcpy(p1.name, "spring");
  Person * p2 = malloc(sizeof(Person));
  p2 -> name = malloc(sizeof(char) * 80);
  strcpy(p2 -> name, "ece");
  Person * p3 = & p1;
  g(& p3, p2, p1);
  printf("p1.name = %s, p2->name = %s, p3->name = %s\n",
         p1.name, p2 -> name, p3 -> name);
  /* do not worry about memory leak */
  return 0;
}
```

After running the code, what are the values of

- (1 point) `* y`
- (1 point) `z`
- (2 points) `p2 -> name`
- (2 points) `p3 -> name`

2 Structure (7 points, outcome 2)

Define the structure for `Student` and the associated functions. Do **not** assume a maximum length of a student's name.

```
/* fill the parts marked by ===== */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
/* ==== Create a structure for Student (1 point)
 * Each Student object has two attributes called name and ID.
 * name's type is char * (i.e. string)
 * do not assume the maximum length of name
 * ID's type is int ==== */
```

```
/* ==== Write the constructor (1 point)
 * i is the value for the student's ID
 * n is the string as the student's name
 * The function has to copy n to the Student object's name
 * This function returns a pointer ==== */
Student * Student_construct(int i, char * n)
{









}
/* ==== Write the destructor (1 point) ==== */
void Student_destruct(Student * s)
{
}
/* ==== Write the print function (1 point)
 * print the two attributes ==== */
void Student_print(Student * s)
{







}
int main(int argc, char * argv[])
{
  /* ==== Create one Student object s1 using 123 as the ID and
   * "Amy Smith" as the name (1 point) ==== */









  /* ==== Print s1 using the print function (1 point) ==== */









  /* ==== Destroy s1 and release memory (1 point) ==== */






  return 0;
}
```

3 Structure-1 (7 points, outcome 2)

Define the structure for `Vector` and the associated functions.

```
/* fill the parts marked by ===== */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
/* ==== Create a structure for Vector (1 point)
 * Each Vector object has two attributes called x and y.
 * Both attributes are int. ==== */








/* ==== Write the constructor (1 point) ==== */
/* the two arguments are for x and y respectively */
Vector Vector_construct(int a, int b)
{






}
/* ==== Write the add function (1 point) ==== */
/* return v1 + v2 */
Vector Vector_add(Vector v1, Vector v2)
{






}
/* ==== Write the print function (0.5 point) ==== */
/* print the two attributes */
void Vector_print(Vector v)
{
}
int main(int argc, char * argv[])
{
  /* ==== Create one Vector object v1 using 1.2 and 3.4 as
   * the arguments (1 point) ==== */






  /* ==== Create another Vector object v2 using 7.9 and -6.5 as
   * the arguments (1 point) ==== */






  /* ==== Create Vector v3 using the add function (1 point) ==== */






  /* ==== Print v3 using the print function (0.5 point) ==== */






  return 0;
}
```

4 Complexity (bonus 1 point)

What is `c1 + c2` for matrix multiplication expressed in the following code? Assume each matrix has n rows and n columns. Write your answer using n. You can ignore lower-degree terms and coefficients; for example, $2n^4 + 6n$ can be written as n^4.

```
/*
 * Assume each matrix is an n x n array
 * C = A x B
 */
void Matrix_multiply(int * * A, int * * B, int * * C, int n)
{
  int i, j, k;
  /* initialize C elements */
  int c1 = 0;
  int c2 = 0;
  for (i = 0; i < n; i ++)
    {
      for (j = 0; j < n; j++)
        {
          c1 ++;
          C[i][j] = 0;
        }
    }
  for (i = 0; i < n; i ++)
    {
      for (j = 0; j < n; j++)
        {
          for (k = 0; k < n; k ++)
            {
              c2 ++;
              C[i][j] += A[i][k] * B[k][j];
            }
        }
    }

}
```