EECE.2160: ECE Application Programming

Spring 2017

Exam 3
May 5, 2017

Name: _____

Section (circle 1): **201** *(Dr. Li, MWF 8-8:50)* **202** *(Dr. Geiger, MWF 12-12:50)*

For this exam, you may use only one 8.5" x 11" double-sided page of notes. All electronic devices (e.g., calculators, cell phones, tablets) are prohibited. If you have a cell phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 3 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

Please read each question carefully before you answer. In particular, note that:

- Question 3 has three parts, but you are only required to complete two of the three parts.
  - You may complete all 3 parts for up to 10 points of extra credit. If you do, **please indicate which part is extra—we will assume part (c) is if you mark nothing.**
- For each part of Question 3, you must complete a short function. We have provided comments to describe what your function should do and written some of the code for you.
  - Each function contains both partially written lines (for example, a `printf()` call missing the format string and expressions) and blank spaces in which you must write additional code. **You must write all code required to make each function work as described—do not simply fill in the blank lines.**
  - Each given test case is an example of how the function should behave in one specific case—**it does not cover all possible results of using that function.**
  - Each of these functions works with one or more structures. **You can find the structure definitions on the extra sheet provided with the exam.**
- You can solve each part of Question 3 using only the variables that have been declared, but you may declare and use other variables if you want.

You will have 3 hours to complete this exam.

Q1: Multiple choice	/ 20
Q2: Dynamic memory allocation	/ 40
Q3: Structures	/ 40
TOTAL SCORE	/ 100
EXTRA CREDIT	/ 10

1. (20 points, 4 points per part) ***Multiple choice***

For each of the multiple choice questions below, clearly indicate your response by circling or underlining the one choice you think best answers the question.

a. Given an input text file "input.txt" that contains the following information:

```
10 12.3 33.5 end line
```

Which of the following code sequences can be used to read ALL the information from this file? Please note some code sequences may generate compilation errors.

i.
```
FILE *fp;
double d1, d2, d3;
char str[10];
fp = fopen("input.txt", "r");
fprintf (fp, "%lf %lf %lf %s", d1, d2, d3, str);
fclose(fp);
```

ii.
```
FILE *fp;
double d1, d2, d3;
char str[10];
fp = fopen("input.txt", "r");
fscanf (fp, "%lf %lf %lf %s", d1, d2, d3, str);
fclose(fp);
```

iii.
```
FILE *fp;
double d1, d2, d3;
char str[10];
fp = fopen("input.txt", "r");
fscanf (fp, "%lf %lf %lf %s", &d1, &d2, &d3, &str);
fclose(fp);
```

iv.
```
FILE *fp;
double d1, d2, d3;
char str1[10], str2[10];
fp = fopen("input.txt", "r");
fscanf (fp, "%lf %lf %lf %s %s",
        &d1, &d2, &d3, &str1, &str2);
fclose(fp);
```

v.
```
FILE *fp;
double d1, d2, d3;
char str1[10], str2[10];
fp = fopen("input.txt", "r");
fscanf (fp, "%lf %lf %lf %s %s",
        &d1, &d2, &d3, str1, str2);
fclose(fp);
```

1 (continued)

b. You have a program that contains an array declared as:

```
int arr[10];
```

And you have a binary file "input.bin" that contains data. Which of the following code sequences would read enough data from the binary file to fill the entire array arr?

A.
```
FILE *fp = fopen("input.bin", "rb");
fread(arr, sizeof(int), 10, fp);
fclose(fp);
```

B.
```
FILE *fp = fopen("input.bin", "rb");
fread(arr, sizeof(int) * 10, 1, fp);
fclose(fp);
```

C.
```
FILE *fp = fopen("input.bin", "rb");
fwrite(arr, sizeof(int), 10, fp);
fclose(fp);
```

D.
```
FILE *fp = fopen("input.bin", "rb");
int i;
for (i = 0; i < 10; i++) {
    fscanf(fp, "%d", &arr[i]);
}
fclose(fp);
```

i. Only A

ii. A and B

iii. A and D

iv. A, B and C

v. A, B and D

1 (continued)

c. Assume a program user types the following at the command line: `E2as5y 7Exa9m`

Assume your program contains a single character `c` and a string `str` as follows:

```
char c;
char str[20];
```

Which of the following code sequences would extract the string `"EasyExam"` from `stdin` and store it in the string `str`? Remember the function `isalpha` used in Program 6:

`int isalpha(int ch)`: This function returns a non-zero value if `ch` is a letter of the alphabet, and zero otherwise. For example, `isalpha('Q')` returns 1; `isalpha('3')` returns 0.

i.
```
int i = 0;
while ((c = getchar()) != '\n'){
   if (isalpha(c) != 0)
      str[i++] = c;
}
str[i] = 0;    // Remember, ASCII value 0 = '\0'
```

ii.
```
int i = 0;
FILE *fp = fopen("input.txt", "r");
while ((c = fgetc(fp)) != '\n'){
   if (isalpha(c))
      str[i++] = c;
}
str[i] = 0;
```

iii.
```
int i = 0;
while ((c = getchar()) != '\n'){
   if (isalpha(c) == 0)
      ungetc (c);
   else
      str[i++] = c;
}
str[i] = 0;
```

iv.
```
int i = 0;
fgets(str, 20, stdin);
while ((c = str[i]) != '\0'){
   if (isalpha(c) != 0)
      printf("%c", c);
}
str[j] = 0;
```

1 (continued)

d. You have two unsigned integers defined as follows:

```
unsigned int a = 0x9876ABCD;
unsigned int b = 15;
```

Which of the following assignments can set variable `a` equal to `0xBC` (extract bit 4-11)?

A. `a = (a & 0x00000FF0) >> 4;`

B. `a = (~b & a) >> 4;`

C. `a = (a >> 4) & 0x000000FF;`

D. `a = (~0x00000FF0 | a) >> 4;`

i. Only A

ii. A and B

iii. A and C

iv. B and C

v. C and D

e. Circle one (or more) of the choices below that you feel best "answers" this "question."

i. "Thanks for the free points."

ii. "I don't REALLY have to answer the last two questions, do I?"

iii. "This is the best final exam I've taken today."

iv. None of the above.

2. (40 points) ***Dynamic memory allocation***

For each short program shown below, list the output exactly as it will appear on the screen. Be sure to clearly indicate spaces between characters when necessary. Assume all necessary libraries are included.

You may use the available space to show your work as well as the output; just be sure to clearly mark where you show the output so that I can easily recognize your final answer.

a. (12 points)

```
int main () {
    int arr[5] = {8, 4, 2, 1};
    int *ptr;
    int i;

    ptr = (int *)malloc(sizeof(int) * 5);
    for (i = 0; i < 5; i++) {
        ptr[i] = arr[i] * 4;
        printf("%d ", ptr[i]);
    }
    printf("\n");

    ptr = (int *)realloc(ptr, sizeof(int) * 7);
    for (i = 0; i < 7; i++) {
        ptr[i + 1] = ptr[i] / 2;
        printf("%d ", ptr[i]);
    }
    printf("\n");

    free(ptr);
    return 0;
}
```

2 (continued)

b. (14 points) ***See the provided extra sheet for the `MyStruct` and `myPrint()` definitions.***

```
int main () {
    int i;
    MyStruct var1 = {2, 2.1};
    MyStruct var2 = {3, 3.1};
    MyStruct * arr;

    myPrint(&var1);
    myPrint(&var2);

    arr = (MyStruct *)calloc(4, sizeof(MyStruct));
    for(i = 0; i < 3; i++) {
        arr[i+1].a = arr[i].a + var1.a;
        arr[i+1].b = arr[i].b + var1.b;
        myPrint(&arr[i]);
    }
    myPrint(&arr[i]);

    arr = (MyStruct *)realloc(arr, sizeof(MyStruct));
    myPrint(&arr[0]);

    free(arr);
    return 0;
}
```

2 (continued)

c. (14 points) ***See the provided extra sheet for the `LLnode`, `func1()`, and `func2()` definitions***

The following program is based on the sorted linked list lecture. Assume we already have the `addSortedNode` function defined, and the function prototype is described as follows:

```
// Keep list sorted in ascending order and add integer v;
// return pointer to start of list
LLnode *addSortedNode(LLnode *list, int v);

int main () {
    int i;
    int arr [6] = {5, 3, 6, 3, 1, 5};
    LLnode * list = NULL;

    for (i = 0; i < 6; i++) {
        list = addSortedNode(list, arr[i]);
    }
    func1(list);
    func1( func2(list) );

    free(list);
    return 0;
}
```

3. (40 points, 20 per part) ***Structures***

For each part of this problem, you are given a short function to complete. **CHOOSE ANY TWO OF THE THREE PARTS** and fill in the spaces provided with appropriate code.

You may complete all three parts for up to 10 points of extra credit, but must clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.

Remember, you must write all code required to make each function work as described—**do not assume you can simply fill in the blank lines and get full credit.**

Also, remember that each example provided is only applicable in one specific case—**it does not cover all possible results of using that function.**

In order to allow plenty of space to solve each problem, this page is essentially just a "cover sheet" for Question 3—**the actual problems start on the next page**.

Each of these functions works with one or more structures. **You can find the structure definitions on the extra sheet provided with the exam.**

3 (continued)

a. `void matrixMult(Matrix *m1, Matrix *m2, Matrix *p);`

This function multiplies two `Matrix` structures pointed to by `m1` and `m2` and stores the product (and its dimensions) in the structure pointed to by `p`. Matrix multiplication works as follows:

- The number of columns in the first matrix must match the number of rows in the second
- The product has the same number of rows as the first matrix and the same number of columns as the second matrix
- The value in row `i` and column `j` of the product is the dot product of row `i` from matrix 1 and column `j` from matrix 2. To calculate a dot product, multiply matching members and add the results.

The example below shows the 2x2 product of multiplying a 2x3 matrix and a 3x2 matrix:

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \times \begin{bmatrix}3 & 2\\ 6 & 4\\ 9 & 8\end{bmatrix} = \begin{bmatrix}1\times3+2\times6+3\times9 & 1\times2+2\times4+3\times8\\ 4\times3+5\times6+6\times9 & 4\times2+5\times4+6\times8\end{bmatrix} = \begin{bmatrix}42 & 34\\ 96 & 76\end{bmatrix}$$

```
void matrixMult(Matrix *m1, Matrix *m2, Matrix *p) {
   int i, j, k;      // Loop indexes

   // End function if matrices can't be multiplied

   if (_____________________________________________________________)
      return;

   // Set dimensions (rows & columns) of product




   // Perform actual multiplication as described above

   for (_____________________________________________________) {

      for (_____________________________________________________) {



         for (_____________________________________________________) {









      }
   }
}
```

3 (continued)
b. `int maxVol(Box list[], int n);`

This function takes two arguments: (1) `list` is an array of `Box` structures, and (2) `n` is the number of structures in the array `list`. The function returns the index of the structure representing the box with the greatest volume. For example, given:

```
Box arr[4] = { {1, 3, 5}, {9, 9, 9}, {4, 3, 2}, {2, 2, 2} };
```

calling `maxVol(arr, 4)` would return 1, as `arr[1]` has the greatest volume (`arr[0]` has volume 15, `arr[1]` has volume 729, `arr[2]` has volume 24, and `arr[3]` has volume 8).

```
int maxVol(Box list[], int n) {
   int i;            // Loop index
   int maxI;         // Index of largest prism
   double maxVol;    // Volume of largest prism

   // Initialize variables as needed









   // Go through list; update max variables if larger box found

   for (_____________________________________________________) {

      if (_____________________________________________________) {









      }
   }

   // Return index of box with greatest volume

   return ____________________________________;
}
```

3 (continued)

c. `void inventory(AutoPart list[], int n, char carMake[]);`

This function takes three arguments: (1) `list`, an array of `AutoPart` structures, (2) `n`, the number of structures in the list, and (3) `carMake`, a string corresponding to a brand of car.

Your solution should go through the entire list and print the name, make, and year of all `AutoPart` structures that (1) are in stock (i.e., `inStock` is "true") and (2) have a make (contained in the nested structure called `type`) that matches `carMake`. For example, given:

```
AutoPart partList[] = { {"axle", {"Ford", 2000}, 1},
                        {"tire", {"Ford", 2010}, 0},
                        {"engine", {"Chevy", 2005}, 1},
                        {"mirror", {"Ford", 1978}, 1},
                        {"lighter", {"Chevy", 1957}, 0} };
```

the function call `inventory(partList, 5, "Chevy")` would print:

```
engine
Chevy
2005
```

```
void inventory(AutoPart list[], int n, char carMake[]) {
   int i;       // Loop index

   // Go through entire list

   for (_____________________________________________________) {

      // If current structure represents part that is in stock
      //   and make is correct, print contents of structure

      if (_____________________________________________________

          _____________________________________________________) {









      }
   }
}
```