EECE.2160: ECE Application Programming

Fall 2016

Exam 3 Solution

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. Which of the following choices attempts to open a file for writing and, if the file cannot be opened, ensures all output intended for the file will be printed to the screen instead?

i.
```
FILE *fp = fopen("myfile.txt", "r");
if (fp == NULL)
   fp = stdin;
```

ii.
```
FILE *fp = fopen("myfile.txt", "r");
if (fp == NULL)
   fp = stdout;
```

iii.
```
FILE *fp = fopen("myfile.txt", "w");
if (fp == NULL)
   fp = stdin;
```

iv.
```
FILE *fp = fopen("myfile.txt", "w");
if (fp == NULL)
   fp = stdout;
```

v. None of the above

1 (continued)

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

```
int arr[40];
```

Which of the following code snippets would correctly read the contents of this array from a file?

i.
```
FILE *fp = fopen("input.txt", "rb");
fread(arr, sizeof(int), 40, fp);
```

ii.
```
FILE *fp = fopen("input.txt", "rb");
fread(fp, sizeof(int), 40, arr);
```

iii.
```
FILE *fp = fopen("input.txt", "r");
fscanf(fp, "%lf", arr);
```

iv.
```
FILE *fp = fopen("input.txt", "rb");
fwrite(arr, sizeof(double), 40, fp);
```

1 (continued)

c. Assume the pointer `LLnode *first` points to the first node of a linked list (see the structure definitions sheet for the `LLnode` definition). Which of the following code sequences counts the number of nodes in the list and stores that value in a variable `count`?

i.
```
unsigned int count = 0;
LLnode *p = first;
while (p == NULL) {
   p = p->next;
   count++;
}
```

ii.
```
unsigned int count = 0;
LLnode *p = first;
while (p != NULL) {
   p = p->next;
}
```

iii.
```
unsigned int count = 0;
LLnode *p = first;
while (p != NULL) {
   p = p->next;
   count++;
}
```

iv.
```
unsigned int count = 0;
LLnode *p = first;
while (p->next != NULL) {
   p = p->next;
   count++;
}
```

1 (continued)

d. Assume the pointer `LLnode *first` points to the first node of a linked list (see the structure definitions sheet for the `LLnode` definition). Which of the following pieces of code modifies the list so that it is circular? (In a circular linked list, rather than the "`next`" pointer in the last node being NULL, the "`next`" pointer in the last node points to the first node.)

i.
```
last->next = first;
```

ii.
```
LLnode *p = first;
while (p != NULL)
   p = p->next;
p->next = first;
```

iii.
```
LLnode *p = first;
while (p->next != NULL)
   p = p->next;
p->next = first;
```

iv.
```
LLnode *p = first;
p->next = first;
```

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.

All of the above are "correct."

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. (13 points)

```
int main() {
   int len[] = {5, 3, 6, 2};
   int *p1 = NULL;
   int i, j;

   for (i = 0; i < 4; i++) {
      if (p1 == NULL)
         p1 = (int *)calloc(len[i], sizeof(int));
      else
         p1 = (int *)realloc(p1, len[i] * sizeof(int));

      for (j = 0; j < len[i]; j++) {
         p1[j] = len[j % 4] / (i + 1);
         printf("%d ", p1[j]);
      }
      printf("\n");
   }

   free(p1);
   return 0;
}
```

OUTPUT:

5 3 6 2 5
2 1 3
1 1 2 0 1 1
1 0

2 (continued)
b. (13 points) Remember: `tolower('A') = 'a'; tolower('b') = 'b'`

```
int main() {
    char *str;
    char test1[] = "ECE Application Programming";
    char test2[] = "xcellent";

    str = (char *)malloc((strlen(test1)+1) * sizeof(char));
    strcpy(str, test1);
    printf("%s\n", str);

    str[1] = '\0';
    str = (char *)realloc(str,
          (strlen(test1) + strlen(test2) + 1) * sizeof(char));
    strcat(str, test2);
    printf("%s\n", str);

    str = (char *)realloc(str, 5 * sizeof(char));
    str[0] = tolower(str[0]);
    str[2] = str[2] - 2;      // Hint: think about ASCII values,
    str[3] = str[4] + 1;      //   letters, & alphabetical order
    str[4] = '\0';
    printf("%s\n", str);

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

OUTPUT:
ECE Application Programming
Excellent
exam

2 (continued)

c. (14 points) *See the provided extra sheet for the `LLnode` structure definition, as well as the `addNode()` and `printList()` function definitions*

```
LLnode *func(LLnode *list) {
   LLnode *prev = NULL;
   LLnode *cur  = list;
   LLnode *next = NULL;
   while (cur != NULL) {
      next = cur->next;
      cur->next = prev;
      prev = cur;
      cur = next;
   }
   return prev;
}

void main() {
   LLnode *head = NULL;
   char myStr[] = "Exam 3";
   char temp;
   int i = 0;
   while ( (temp = myStr[i++]) != 0 ) {
      if (temp != ' '){
         printf("%c\n", temp);
         head = addNode(head, temp);
      }
   }
   printList(head);
   printList(func(head));
}
```

OUTPUT:

```
E
x
a
m
3
3maxEExam3
```

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 (m1->nc != m2->nr)
      return;

   // Set dimensions (rows & columns) of product
   p->nr = m1->nr;
   p->nc = m2->nc;

   // Perform actual multiplication as described above
   for (i = 0; i < p->nr; i++) {
      for (j = 0; j < p->nc; j++) {
         p->mat[i][j] = 0;
         for (k = 0; k < m1->nc; k++) {
            p->mat[i][j] += m1->mat[i][k] * m2->mat[k][j];
      }
   }
}
```

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

This function takes as arguments an array of `Box` structures, `list`, as well as the number of structures in the list, `n`. 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} };
```

the function call `maxVol(arr, 4)` would return 1, as `arr[1]` has the greatest volume (729, which is greater than the volumes of `arr[0]` (volume 15), `arr[2]` (24), and `arr[3]` (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
   maxI = 0;
   maxVol = list[0].W * list[0].L * list[0].H;

   // Go through list; update max variables if larger box found
   for (i = 1; i < n; i++) {
      if (maxVol < list[i].W * list[i].L * list[i].H) {
         maxVol = list[i].W * list[i].L * list[i].H;
         maxI = i;
      }
   }

   // Return index of box with greatest volume
   return maxI;
}
```

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

This function takes as arguments an array of `AutoPart` structures, `list`, as well as the number of structures in the list, `n`, and a string, `carMake`.

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 (i = 0; i < n; i++) {

      // If current structure represents part that is in stock
      //   and make is correct, print contents of structure
      if (list[i].inStock &&
          (strcmp(list[i].type.make, carMake) == 0) {
           printf("%s\n", list[i].name);
           printf("%s\n", list[i].type.make);
           printf("%d\n\n", list[i].type.year);
      }
   }
}
```