EECE.2160: ECE Application Programming

Summer 2016

Lecture 11: Key Questions
June 15, 2016

For today's exercise, you will complete the following functions that work with the structures `Name` and `StudentInfo`. The structure definitions are listed below:

```
typedef struct {
    char first[50];
    char middle;
    char last[50];
} Name;

typedef struct {
    Name sname;
    unsigned int ID;
    double GPA;
} StudentInfo;
```

The function descriptions are as follows:

For the `Name` structure:

- **`void printName(Name *n)`**: Print the name pointed to by `n`, using format `<first> <middle>. <last>`
- **`void readName(Name *n)`**: Prompt for and read a first, middle, and last name, and store them in the structure pointed to by `n`

For the `StudentInfo` structure:

- **`void printStudent(StudentInfo *s)`**: Print information about the student pointed to by `s`
- **`void readStudent(StudentInfo *s):`** Prompt for and read information into the student pointed to by `s`
- **`void printList(StudentInfo list[], int n):`** Print the contents of an array list that contains `n` `StudentInfo` structures
- **`int findByLName(StudentInfo list[], int n, char lname[]):`** Search for the student with last name `lname` in the array `list`. Return the index of the structure containing that last name, or -1 if not found
- **`int findByID(StudentInfo list[], int n, unsigned int sID)`**: Search for the student with ID # `sID` in the array `list`. Return the index of the structure containing that last name, or -1 if not found

From Name.c:

```
// Print contents of Name struct
void printName(Name *n) {




}

// Read information into existing Name
void readName(Name *n) {







}
```

From StudentInfo.c:

```
// Print information about student
void printStudent(StudentInfo *s) {




}

// Reads student information into existing structure
void readStudent(StudentInfo *s) {









}
```

From StudentInfo.c (continued):

```
// Print list of students
void printList(StudentInfo list[], int n) {









}

// Find student in list, based on last name
// Returns index if student found, -1 otherwise
int findByLName(StudentInfo list[], int n, char lname[]) {









}
```

From StudentInfo.c (continued):

```
// Find student in list, based on ID #
// Returns index if student found, -1 otherwise
int findByID(StudentInfo list[], int n, unsigned int sID) {









}
```

Dynamic memory allocation questions:

1. Explain the `malloc()` function.

2. Explain the use of type casting, and why it is necessary with the allocation functions.

3. Explain the `calloc()` function.

4. Explain the `realloc()` function.

5. Explain how free() is used to deallocate memory.

6. **Example:** What does the following program print?

```
void main() {
   int *arr;
   int n, i;

   n = 7;
   arr = (int *)calloc(n, sizeof(int));
   for (i = 0; i < n; i++)
      printf("%d ", arr[i]);
   printf("\n");

   n = 3;
   arr = (int *)realloc(arr, n * sizeof(int));
   for (i = 0; i < n; i++) {
      arr[i] = i * i;
      printf("%d ", arr[i]);
   }

   n = 6;
   arr = (int *)realloc(arr, n * sizeof(int));
   for (i = 0; i < n; i++) {
      arr[i] = 10 - i;
      printf("%d ", arr[i]);
   }

   free(arr);
}
```

7. What are the common pitfalls of dynamic memory allocation?

8. Explain how to use dynamic memory allocation with strings.

9. Explain how to use dynamic memory allocation with two-dimensional arrays.

10. **Example:** Write each of the following functions:

a. **`char *readLine():`** Read a line of data from the standard input, store that data in a dynamically allocated string, and return the string (as a **`char *`**)
Hint: Read the data one character at a time and repeatedly reallocate space in string

b. **`int **make2DArray(int total, int nR);`** Given the total number of values and number of rows to be stored in a two-dimensional array, determine the appropriate number of columns, allocate the array, and return its starting address
Note: if **nR** does not divide evenly into **total**, round up. In other words, an array with 30 values and 4 rows should have 8 columns, even though 30 / 4 = 7.5