

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 `SINew`. 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;
} SINew;
```

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(SINew *s):** Print information about the student pointed to by `s`
- **void readStudent(SINew *s):** Prompt for and read information into the student pointed to by `s`
- **void printList(SINew list[], int n):** Print the contents of an array `list` that contains `n` `SINew` structures
- **int findByName(SINew 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(SINew 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 SINew.c:

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

}

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

}

}
```

From SINew.c (continued):

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

}

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

}

}
```

From SInew.c (continued):

```
// Find student in list, based on ID #  
// Returns index if student found, -1 otherwise  
int findByID(SInew 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.

