CSCI 136: Data Structures and Advanced Programming

Lecture 19-1

Resubmissions

Instructor: Dan Barowy

Williams

Outline

1. Resubmission procedure
2. Ordered structures

Resubmission procedure

Resubmission procedure

Remember: the goal of this course is mastery.

Resubmission procedure

Allows you to earn **up to 50% of the lost points**.

E.g., **if you got a 50%** on the midterm, **you can get a 75%** on resubmission.

Midterm is 20% of your final grade.
This is worth doing!

Resubmission procedure

1. You have **until the end of the semester**.
2. Resubmission **must include both** the **original work** and the **new submission**.
3. Must be accompanied by an **explanation document**, written in plain English.

Resubmission procedure

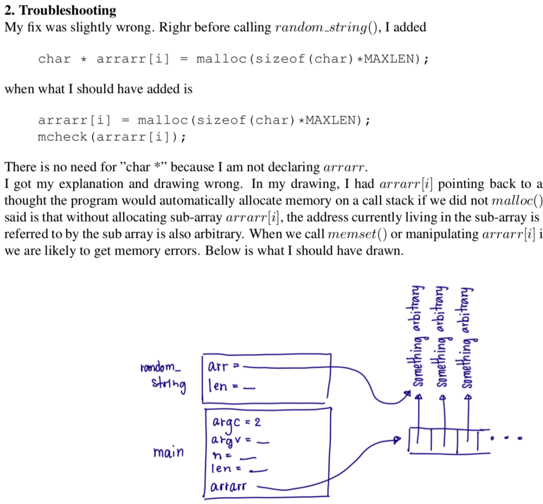
Explanation document **must identify**:

1. **What** the mistake is.
2. **How** you fixed the mistake.
3. **Why** the new version is correct.

Resubmission procedure

Please submit this **electronically** (email is fine).

Resubmission procedure Sample:



2. Troubleshooting

My fix was slightly wrong. Righr before calling $random\_string()$, I added

```
    char * arrarr[i] = malloc(sizeof(char)*MAXLEN);
```

when what I should have added is

```
    arrarr[i] = malloc(sizeof(char)*MAXLEN);
    mcheck(arrarr[i]);
```

There is no need for "char *" because I am not declaring $arrarr$.

I got my explanation and drawing wrong. In my drawing, I had $arrarr[i]$ pointing back to a call stack because I thought the program would automatically allocate memory on a call stack if we did not $malloc()$. What I should have said is that without allocating sub-array $arrarr[i]$, the address currently living in the sub-array is arbitrary so the value referred to by the sub array is also arbitrary. When we call $memset()$ or manipulating $arrarr[i]$ in $random\_string()$, we are likely to get memory errors. Below is what I should have drawn.

Recap & Next Class

This lecture:

Resubmission procedure

Next lecture:

Ordered structures