Lab 9

Thursday, November 16, 2023

You will get 1 point for attending the lab, making an effort to work on the problems, and **discussing your work with a lab instructor during the lab**. If you finish during the lab, demonstrate your working code to a lab instructor and they will give you the full 2 points. If you do not finish during the lab, discuss what you've done with a lab instructor and they will give you an opportunity to finish outside of the lab and attend office hours to demonstrate your solution.

Practice Problems. In the previous lab you used structures for recording student information of students with names at most 19 characters. In this lab, we allow names of arbitrary length by using dynamic memory and the following updated student structure definition:

```
struct student {
  int id;          // student ID
  char * name;     // pointer to the first name of the student (on the heap)
};
```

Complete an implementation of the following functions that creates and frees student records.

```
// create_student(id, name) creates a new student record with given id and name;
//   allocates memory to store structure and name (must free with free_student)
//   and returns NULL if memory allocation fails
// requires: name points to a valid string
struct student * create_student(int id, char * name);

// free_student(s) frees the memory associated with the student record s
// requires: s is a student record created using dynamic memory allocation
void free_student(struct student * s);
```

Create at least 3 student records with `create_student`, including one that contains your own name and student ID. Test that the `id` and `name` are correct using `assert` and `strcmp`, and then release the allocated memory using `free_student`. You do not need to test the case when the memory allocation fails. What is the running time of `create_student` and `free_student` applied to a student with a name of length n?

Marked Problem. Complete an implementation of a function that changes the name of a given student. The function should use `malloc` or `realloc` to allocate enough memory to store the new name. If the memory allocation fails then return `false` and do not change the student's name; otherwise return `true` if the student's name was successfully renamed.

```
// change_name(s, new_name) renames the student s to have the name given by
//   new_name; allocates memory to hold new_name (caller must free)
//   returns true when the name was successfully changed
// requires: s points to a valid student and new_name points to a valid string
bool change_name(struct student * s, char * new_name);
```

Test that `change_name` works correctly using `assert` and `strcmp` when the new name is longer or shorter than the original name. **Ensure your implementation does not leak memory.**