

# SMALL PROGRAM 8

---

COP3223C Introduction to Programming with C  
Dr. Andrew Steinberg

Fall 2022

## Due Date

The assignment is due on December 2nd at 11:59pm EST via Webcourses. **Do not email the professor or TAs your submissions as they will not be accepted!** This assignment is accepted late up to 24 hours with a penalty. Please see the syllabus for more information on this. Make sure to submit on time to get potential full credit. Make sure to also take into consideration the uploading time. In the past, students who are working last minute on the assignment sometimes run into uploading issues where their Internet may run slow, resulting in late submissions. The timestamp Webcourses uses for your submission will be applied and will be the final say. Please do not email the instructor or TAs saying your Internet was running slow. If the time is off by a second of the due date, then the assignment is considered late. Plan accordingly!

## Important! Read Carefully! There is new info here!

This assignment contains a set of problems that are to be completed in **one C file**. For this assignment, you are provided a skeleton file (with some code) in the assignment page called `smallprogram8skeleton.c`. The file has the following definition of a typedef struct.

---

```
typedef struct{
    char * fname; //first name
    char * lname; //last name
    char * show; //favorite show
}record_t;
```

---

The typedef struct has three members that are all character pointers to a respective heap (dynamic string). In this assignment, you are managing a dynamic array of the typedef struct. The file will compile on Eustis and run, however it is not fully completed. In this assignment you are going to fill in the remainder code. The remainder code will involve implementing 4 user defined functions for each problem. The file must be named *smallprogram8\_lastname\_firstname.c*, where lastname and firstname is your last and first name (as registered in webcourses). For example Dr. Steinberg's file would be named *smallprogram8\_Steinberg\_Andrew.c*. Make sure to include the underscore character `_`. If your file is not named properly, points will be deducted. The script the graders use will pull your name from the file name properly. It is imperative you follow these steps so you can get your points!

## Testing on Eustis

It is your responsibility to test your code on Eustis. If you submit your assignment without testing on Eustis, you risk points being deducted for things that may behave differently on your operating system. Remember, you cannot dispute grades if your code didn't work properly on Eustis all because it worked on your machine. The Eustis environment gives the final say. Plan accordingly to test on Eustis!!

## The Python Script File! Read Carefully!

A python script has been provided for you to test your code with a sample output of Dr. Steinberg's solution. This script will check to make sure your output matches exactly with Dr. Steinberg's solution file as the graders are using this to grade your assignments. The script removes leading and trailing white space, but not white space in the actual text. If there is anything off with the output (including the expected answer), the script will say your output is not correct. This includes your output producing the correct answer, however there is something off with the output display. The script does not point directly where your mistake(s) are in the code. It will only produce a success or unsuccessful output message as a whole. If you get an unsuccessful output my suggestion is to look at the sample solution text file provided to see what is different from your answer and Dr. Steinberg's when comparing. If you have extra white space or new lines or even just missing a space/new line, you will lose points that won't be changed!

Make sure you place the python script in the same directory as your C file. You can use the ls command to check to see if the following items are in the directory.

---

```
ls
```

---

You should have these three files in your directory.

1. Your C File.
2. The Python Script File
3. Sample Solution Text File

If you have these three files. You are ready to run the script. Use the following command to test your code with Dr. Steinberg's provided solution sample.

---

```
python3 sp8test.py
```

---

If the script says your output is incorrect, checkout the sample text file that was generated (a new text file will be created from the script that contains YOUR output). If your numbers are off or different from Dr. Steinberg's, then that means there is something not right with code's logic and calculating the answer. However if your numbers match with Dr. Steinberg's solution,

then that means there is extra/missing white space or newlines detected. Compare the text file generated by the script with solution text file line by line to find the missing/extra white space or newlines. Once you believe you found the error, rerun the script to see if the output matches.

Now, you are probably wondering how the graders will know if you have memory leaks in your code. The script the graders are using involves valgrind which checks for memory leaks. You will learn more about this in CS1, but it's good to be exposed to this tool and understand how it works. In your test script, valgrind will be executed with your code. The script will check to see if you have any memory leaks. If your code has no memory leaks, then a message like this will be displayed to the terminal. You may run into getting the correct output with this memory leak message. **Please note that if you submit code that has correct output, but memory leaks will result in point deductions.**

```
Ugh oh... You have memory leaks! Make sure to fix those!
-----
==1445948== Memcheck, a memory error detector
==1445948== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1445948== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==1445948== Command: ./smallprogram8
==1445948==
==1445948==
==1445948== HEAP SUMMARY:
==1445948==   in use at exit: 192 bytes in 1 blocks
==1445948==   total heap usage: 40 allocs, 39 frees, 10,280 bytes allocated
==1445948==
==1445948== 192 bytes in 1 blocks are definitely lost in loss record 1 of 1
==1445948==   at 0x483B7F3: malloc (in /usr/lib/x86_64-linux-gnu/valgrind/vgpreload_memcheck-amd64-linux.so)
==1445948==   by 0x109482: createarray (in /home/net/an785125/COP3223/SmallPrograms/sp8/smallprogram8)
==1445948==   by 0x109857: doubleit (in /home/net/an785125/COP3223/SmallPrograms/sp8/smallprogram8)
==1445948==   by 0x1094F2: insert (in /home/net/an785125/COP3223/SmallPrograms/sp8/smallprogram8)
==1445948==   by 0x109387: main (in /home/net/an785125/COP3223/SmallPrograms/sp8/smallprogram8)
==1445948==
==1445948== LEAK SUMMARY:
==1445948==   definitely lost: 192 bytes in 1 blocks
==1445948==   indirectly lost: 0 bytes in 0 blocks
==1445948==   possibly lost: 0 bytes in 0 blocks
==1445948==   still reachable: 0 bytes in 0 blocks
==1445948==   suppressed: 0 bytes in 0 blocks
==1445948==
==1445948== For lists of detected and suppressed errors, rerun with: -s
==1445948== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
-----
```

Figure 1: Sample of valgrind informing the user that memory leaks have occurred.

## The Rubric

Please see the assignment page for the established rubric on webcourses.

## Comment Header

Make sure you place a comment header at the top of your C file. You will use single line comments to write your name, professor, course, and assignment. For example, Dr. Steinberg's header would be:

---

```
//Andrew Steinberg
//Dr. Steinberg
//COP3223C Section 1
//Small Program 8
```

---

Missing a comment header will result in point deductions!

## The Solution Text File

You are provided a solution file that was created by Dr. Steinberg's Python Script. Now you may notice some strange things about the file. In this assignment, you are going to write statements that involve interacting with the user. Now you are probably wondering from the solution text file where the interaction is happening? The fact is that the Python script handles the interaction. The script creates an input stream that feeds it input. That is why you don't see the input directly in the text file. For example, the text file on line 1 says `Enter a size for the dynamic array: What would you like to do?`. Here this looks like we should be typing input, however the python script has already fed it input. That is why you don't see the values except for the results. In each problem, a screenshot of the C file being executed manually without the Python script shows how it looks on a normal run. Carefully look at the output in the pictures provided. Note: The arrow symbolizes that the text wrapped onto the next line of the pdf file. In the text file itself it is actually one whole line.

Enter a size for the dynamic array: What would you like to do?

- 1: Insert a record
- 2: Display records
- 3: Remove record
- 4: Exit

Enter an option: Insert was selected...

Enter the first name: Enter the last name: Enter favorite show: What would you like  
↩ to do?

- 1: Insert a record
- 2: Display records
- 3: Remove record
- 4: Exit

Enter an option: Insert was selected...

Enter the first name: Enter the last name: Enter favorite show: What would you like  
↩ to do?

- 1: Insert a record
- 2: Display records
- 3: Remove record
- 4: Exit

Enter an option: Insert was selected...

Enter the first name: Enter the last name: Enter favorite show: What would you like  
↩ to do?

- 1: Insert a record
- 2: Display records
- 3: Remove record
- 4: Exit

Enter an option: Display was selected...

-----

```
myarray[0].fname = Kasey
myarray[0].lname = Moss
myarray[0].show = The Lucy Show
```

```
myarray[1].fname = Estrella
myarray[1].lname = Holder
myarray[1].show = Loki
```

```
myarray[2].fname = Zara
myarray[2].lname = Forbes
myarray[2].show = Modern Family
```

-----

What would you like to do?

- 1: Insert a record
- 2: Display records
- 3: Remove record
- 4: Exit

Enter an option: Insert was selected...

Enter the first name: Enter the last name: Enter favorite show: What would you like  
↩ to do?

```

1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Insert was selected...
Array is full...Need to doubleIt...
Enter the first name: Enter the last name: Enter favorite show: What would you like
↵ to do?
1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Display was selected...
-----
myarray[0].fname = Kasey
myarray[0].lname = Moss
myarray[0].show = The Lucy Show

myarray[1].fname = Estrella
myarray[1].lname = Holder
myarray[1].show = Loki

myarray[2].fname = Zara
myarray[2].lname = Forbes
myarray[2].show = Modern Family

myarray[3].fname = Adam
myarray[3].lname = Goldberg
myarray[3].show = The Goldbergs

myarray[4].fname = Terrence
myarray[4].lname = Shaffer
myarray[4].show = The Good Place

-----
What would you like to do?
1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Remove was selected...
Select an index of record to remove...
What would you like to do?
1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Display was selected...
-----
myarray[0].fname = Kasey
myarray[0].lname = Moss

```

```
myarray[0].show = The Lucy Show
```

```
myarray[1].fname = Estrella
```

```
myarray[1].lname = Holder
```

```
myarray[1].show = Loki
```

```
myarray[2].fname = Zara
```

```
myarray[2].lname = Forbes
```

```
myarray[2].show = Modern Family
```

```
myarray[3].fname = Terrence
```

```
myarray[3].lname = Shaffer
```

```
myarray[3].show = The Good Place
```

```
-----
```

```
What would you like to do?
```

```
1: Insert a record
```

```
2: Display records
```

```
3: Remove record
```

```
4: Exit
```

```
Enter an option: Insert was selected...
```

```
Enter the first name: Enter the last name: Enter favorite show: What would you like  
↵ to do?
```

```
1: Insert a record
```

```
2: Display records
```

```
3: Remove record
```

```
4: Exit
```

```
Enter an option: Display was selected...
```

```
-----
```

```
myarray[0].fname = Kasey
```

```
myarray[0].lname = Moss
```

```
myarray[0].show = The Lucy Show
```

```
myarray[1].fname = Estrella
```

```
myarray[1].lname = Holder
```

```
myarray[1].show = Loki
```

```
myarray[2].fname = Zara
```

```
myarray[2].lname = Forbes
```

```
myarray[2].show = Modern Family
```

```
myarray[3].fname = Terrence
```

```
myarray[3].lname = Shaffer
```

```
myarray[3].show = The Good Place
```

```
myarray[4].fname = Harry
```

```
myarray[4].lname = Potter
```

```
myarray[4].show = Stranger Things
```

```
-----
```

```
What would you like to do?
```

```

1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Insert was selected...
Enter the first name: Enter the last name: Enter favorite show: What would you like
↵ to do?
1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Display was selected...
-----
myarray[0].fname = Kasey
myarray[0].lname = Moss
myarray[0].show = The Lucy Show

myarray[1].fname = Estrella
myarray[1].lname = Holder
myarray[1].show = Loki

myarray[2].fname = Zara
myarray[2].lname = Forbes
myarray[2].show = Modern Family

myarray[3].fname = Terrence
myarray[3].lname = Shaffer
myarray[3].show = The Good Place

myarray[4].fname = Harry
myarray[4].lname = Potter
myarray[4].show = Stranger Things

myarray[5].fname = Gary
myarray[5].lname = Foley
myarray[5].show = The Flight Attendant

-----
What would you like to do?
1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Remove was selected...
Select an index of record to remove...
What would you like to do?
1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Display was selected...

```



```

-----
myarray[0].fname = Kasey
myarray[0].lname = Moss
myarray[0].show = The Lucy Show

myarray[1].fname = Estrella
myarray[1].lname = Holder
myarray[1].show = Loki

myarray[2].fname = Zara
myarray[2].lname = Forbes
myarray[2].show = Modern Family

myarray[3].fname = Terrence
myarray[3].lname = Shaffer
myarray[3].show = The Good Place

myarray[4].fname = Harry
myarray[4].lname = Potter
myarray[4].show = Stranger Things

```

```

-----
What would you like to do?
1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Insert was selected...
Enter the first name: Enter the last name: Enter favorite show: What would you like
↵ to do?
1: Insert a record
2: Display records
3: Remove record
4: Exit
Enter an option: Display was selected...

```

```

-----
myarray[0].fname = Kasey
myarray[0].lname = Moss
myarray[0].show = The Lucy Show

myarray[1].fname = Estrella
myarray[1].lname = Holder
myarray[1].show = Loki

myarray[2].fname = Zara
myarray[2].lname = Forbes
myarray[2].show = Modern Family

myarray[3].fname = Terrence
myarray[3].lname = Shaffer
myarray[3].show = The Good Place

```

```
myarray[4].fname = Harry  
myarray[4].lname = Potter  
myarray[4].show = Stranger Things
```

```
myarray[5].fname = Tiffany  
myarray[5].lname = Rivera  
myarray[5].show = Full House
```

-----

What would you like to do?

1: Insert a record

2: Display records

3: Remove record

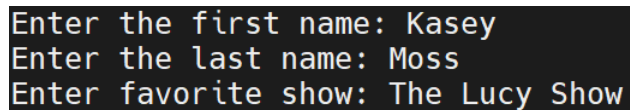
4: Exit

Enter an option: Exiting...

---

## Problem 1

Write the definition of the user-defined function `insert`. The function takes three arguments. The first argument is the reference to a heap of `record_t`. The function will insert a new record into the heap. Make sure to take into consideration if `malloc` doesn't work! Once the heap is updated, the reference is returned. Also, make sure to take into consideration if the array is full! If it is full, you will display `Array is full...Need to doubleIt...` to the terminal and call `doubleIt` which you will implement in problem 2. The following figure shows the output.



```
Enter the first name: Kasey
Enter the last name: Moss
Enter favorite show: The Lucy Show
```

Figure 2: Sample output for problem 1.

## Problem 2

Write a user defined function called `doubleIt` that will double the size of the dynamic array. For example, if the dynamic array initially holds at max 5 elements, then `doubleIt` will create a new dynamic array that holds 10 elements (it will also copy the content from the initial dynamic array). The user defined function takes two arguments in the call. The first argument is a pointer to the heap that contains your initial dynamic array. The second argument is an integer that holds a primitive integer value representing the size. The function returns a reference to the heap that contains the new dynamic array. You cannot use `realloc` or any built in memory function for this question. If `realloc` or any built in memory function is used, then no credit is given for the question. Note you will need to modify the `insert` function bit as `doubleIt` should be called from `insert`. Think about when the function would be invoked! Make sure to take into consideration if `malloc` doesn't work!

## Problem 3

Write a user defined function called `removeRecord` that delete a specific content of the dynamic array. The function takes three arguments. The first argument is the reference to the dynamic array, the second argument represents the current size of the array (meaning valid entries it holds), and the last argument is the index of the value to be remove. Make sure to consider the scenario if the index is out of bounds. The function returns a primitive value that represents the new current size of the dynamic array. Otherwise display the message `Invalid Index` and return the current size if the remove cannot happen due to invalid index. Hint: Use the shift technique for removing.

## Problem 4

Write a user defined function called `freeRecords`. The function takes two arguments. The first argument is the reference to the dynamic array and the second is the current size of the array. The function gives back memory from the heap. Nothing is returned from the function.