EC327 - Spring 2025 - Lab 3

Background

This lab assignment will continue to build our C foundations and will primarily focus on memory management.

Important

Please make sure your code compiles and runs as intended on the engineering grid. Code that does not compile will NOT be graded and will receive a 0.

Submission Instructions

You will submit this assignment as single .zip file with the following name:

<first-name>_<last-name>-lab3.zip

So for example:

ed_solovey-lab3.zip.

Your zip file should contain one file per problem in this assignment. Those files should be named like this:

<first-name>_<last-name>-lab3-<problem-number>.cpp
So for example, my problem one file would be:

ed_solovey-lab3-1.cpp

Each of the problems below will have specific instructions around what the text file for that problem should look like.

Note on Collaboration

You are welcome to talk to your classmates about the assignment and discuss high level ideas. However, all code that you submit must be your own. We will run code similarity tools against your submissions and will reach out with questions if anything is flagged as suspicious.

The closed book exams for this class will mostly cover material very similar to what you do on the homework assignments. The best way to prepare for the exams is to do the assignments on your own and internalize the learnings from that process. Cheating on the assignments will very likely result in you not doing well on the exams.

Problem 1 - Sum of Specific Elements (20 points)

Submission Instructions

Your solution to this problem should contribute a single **cpp** file to your overall **hw3** zip. The **cpp** file should follow the following format:

<first-name>_<last-name>-lab3-1.cpp

So for example, my file would be:

ed_solovey-lab3-1.cpp

Actual Problem

Create a header file called **hw3_problem1.h** with the following contents:

```
#ifndef LAB3_PROBLEM1_H
#define LAB3_PROBLEM1_H

/**
* @param arr An array of [size] integers
* @param size The number of integers stored in array [arr]
* @return The sum of the following for [arr]:
*     * Every element at even index or zero index
*     * Every even element.
*     * The product of the first and last element.
*
* @example 1
* int test[] = {1,2,3};
* labTwo_pZero(test,3) returns 9 as the sum of:
*     * 1+3 (elements at even index)
*     * 2 (even elements)
*     * 3 (first*last element)
*
* @example 2
* int test2[] = {1,2,3,4};
* labTwo_pZero(test,4) returns 14 as the sum of:
```

```
*     * 1+3 (elements at even index)
*     * 2+4 (even elements)
*     * 4 (first*last element)
*/
int sumElements(int* arr, int size);

#endif //LAB3_PROBLEM1_H
```

Your **<first-name>_<last-name>-lab3-1.cpp** should implement the above **sumElements** function.

You can test your implementation from a **main** function or from tests you set up otherwise. Your submission will be tested against multiple inputs and you should convince yourself that your solution works for the general case. This goes for all the problems.

Problem 2 - Reverse String (30 points)

Submission Instructions

Your solution to this problem should contribute a single **cpp** file to your overall **lab3** zip. The **cpp** file should follow the following format:

<first-name>_<last-name>-lab3-2.cpp

So for example, my file would be:

ed_solovey-lab3-2.cpp

Actual Problem

Create a header file called **lab3_problem2.h** with the following contents:

```
#ifndef LAB3_PROBLEM2_H
#define LAB3_PROBLEM2_H



/**
```

```c
* Reverses the given null-terminated string.
*
* This function returns a new string which is the reverse of the input string.
* For example, if the input is "Hello, World!", the function should return
* "!dlroW ,olleH". If the input is NULL, the function should return NULL.
*
* The caller is responsible for freeing the returned string.
*
* @param str A pointer to the null-terminated string to be reversed.
* @return A pointer to a new, dynamically allocated string that is the reverse
*         of the input string, or NULL if memory allocation fails or if str is NULL.
*/
char* reverseString(const char* str);

#endif //LAB3_PROBLEM2_H
```

Your **<first-name>_<last-name>-lab3-2.c** should implement the above **reverseString** function.

Problem 3 - Linked Lists (30 points)

Submission Instructions

Your solution to this problem should contribute a single **cpp** file to your overall **hw3** zip. The **cpp** file should follow the following format:

<first-name>_<last-name>-lab3-3.cpp

So for example, my file would be:

ed_solovey-lab3-3.cpp

Actual Problem

Create a header file called **lab3_problem3.h** with the following contents:

```c
#ifndef HW3_PROBLEM3_H
#define HW3_PROBLEM3_H

#include <stddef.h>
```

```c
#include <stdio.h>

/**
* A node in a linked-list storing integer values.
*/
struct Node {
  int value;
  struct Node* next;
};

/**
* Helper function for debugging the current state of a linked-list
* pointed to by head.
*/
static void printList(struct Node* head) {
  printf("Printing List:\n");
  while (head != NULL) {
      printf("%d -> ", head->value);
      head = head->next;
  }
  printf("NULL\n");
}

/**
* Finds the middle node of a linked list.
*
* For a linked list with an odd number of nodes, this function returns the
* single middle node. For a list with an even number of nodes, it returns
* the node at the start of the second half (i.e. the (n/2 + 1)th node).
*
* If the list is empty (i.e. head is NULL), the function returns NULL.
*
* @param head A pointer to the head of the linked list.
*
* @return A pointer to the middle node of the list, or NULL if the list is empty.
*/
struct Node* findMiddle(struct Node* head);

#endif // HW3_PROBLEM3_MIDDLE_H
```

Your **<first-name>_<last-name>-lab3-3.cpp** should implement the above **findMiddle** function.

Hint: Think about how you could use 2 different pointers.

Problem 4 - Fibonacci Recursion (20 points)

Submission Instructions

Your solution to this problem should contribute a single **cpp** file to your overall **hw3** zip. The **cpp** file should follow the following format:

<first-name>_<last-name>-lab3-4.cpp

So for example, my file would be:

ed_solovey-lab3-4.cpp

Actual Problem

Create a header file called **lab3_problem4.h** with the following contents:

```
#ifndef LAB3_PROBLEM4_H
#define LAB3_PROBLEM4_H



/**
* @param n Position in Fibonacci sequence (0-based)
* @return Fibonacci number at position [n]
* @note Your function must be *recursive* in nature.
* @hint Consider base cases for n=0 and n=1,
*       then combine results from n-1 and n-2
*
* @example
* fibonacci(6) returns 8
* (Sequence: 0, 1, 1, 2, 3, 5, 8)
*/
int fibonacci(int n);

#endif //LAB3_PROBLEM4_H
```

Your **<first-name>_<last-name>-lab3-4.cpp** should implement the above **fibonacci** function.

Requirements:

- Your solution for this problem must use recursion.