Lab 9 - Dynamic Arrays

Due: end of class

OVERVIEW

In this lab, you will process data whose size is not known statically. Thus, you will need to use pointers. To do this, you will represent and manipulate polynomials.

TASKS

A polynomial can be represented through its coefficients. For example, the 4th degree polynomial, $3.5\ x^4 + 7.2\ x^2 + 6x + 5$ can be represented by the array [5.,6.,7.2,0,3.5]. Since the degree of the polynomial is not known in advance, a dynamic array is more appropriate than a static array.

Task 0:

Write a function that inputs the coefficients of a polynomial starting at the highest term. The function should input the degree of the polymonial and the coefficients, using the following dialog (boldface indicates what the user types):

```
Enter polynomial degree: 4
Enter coefficient of term 4: 3.5
Enter coefficient of term 3: 0
Enter coefficient of term 2: 7.2
Enter coefficient of term 1: 6
Enter coefficient of term 0: 5
```

Additionally, the user can enter a end-of-file character (^D on Linux) at any time if all remaining terms are 0.

You are not required to do do input validation on this (though you should if this were not to be done in the lab period).

Your function should create a dynamic array initialized to the input polynomial and return that array.

Task 1:

Write a function that evaluates a polynomial for a given value of x, using the prototype:

```
double eval(double * poly, int degree, double x);
```

You may use the pow function defined in the cmath library if you wish.

Task 2:

First, an important property of polynomials p1 and p2 where p1 has degree greater than p2 and all coefficients are non-negative:

    There exists some x0 such that p1(x)>p2(x) for all x>x0

Although we won't determine this x0 in this lab, we wish to find an x0 that satisfies the first part of the property. Write a main function that does the following:

1. Input two polynomials, p1 and p2. You may have preconditions that p1 and p2 have different degrees, and that all coefficients are non-negative (since its a precondition, you dont need to handle this as an error case)
2. Determine a positive integral x0 such that p1(x0)>p2(x0) (or vice versa). Note that the above property guarantees that such an x0 exists.
3. Output the x0 using the message: "A positive integral x0 such that p1(x0)>p2(x0) is <val>" (or vice versa if p2 is bigger).
4. Delete any dynamic data before returning, to avoid creating garbage. Note that this is considered good practice even though most operating systems will do this automatically when executing return() or exit() (since the C++ standard does not require this cleanup).

You may use any helper functions you want.

Task 3:

Although you may have deleted dynamic data, the data may still be in the heap. To test this, try to access the deleted data by evaluating the corresponding polynomial. State in comments what happens.

Task 4:

Just for fun, make a bounds error in the above program by accessing p1[k] where k is bigger than the polynomial's degree. Explain what happens in comments.

HAND IN

Your 136 instructor will tell you what to hand in and how.

GENERAL COMMENTS FOR ALL PROGRAMS THIS SEMESTER

You should have the following header on all programs:

```
/*
  Author: <name>
  Course: {135,136}
  Instructor: <name>
  Assignment: <title, e.g., "Lab 1">

  This program does ...
*/
```

GRADING

All 135 and 136 programs this semester will be graded on:

- Correctness: Does your program work?
- Testing: Have you generated sufficient and good test data to give reasonable confidence that your program works?
- Structure: Have you structured your code to follow proper software engineering guidelines? This includes readability and maintainability.
- Documentation: How well documented is your code? Good documentation does not repeat the code in English, but explains the point of each code block, highlighting any design decisions and/or tricky implementation details.