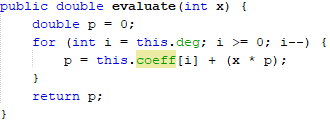
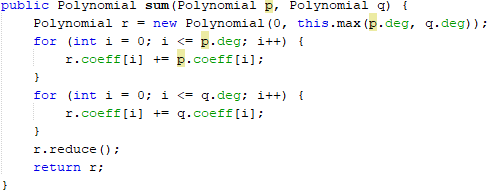
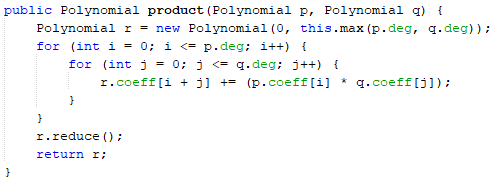
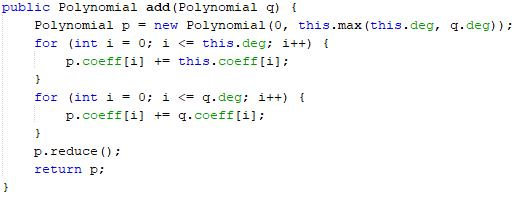
**Please “paste” a copy of your code for the methods discussed at the end of this document.**

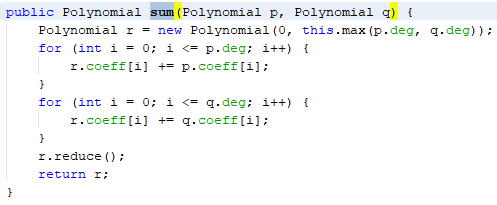
1. **For your array implementation of the Polynomial class, what would be the best value to use for the *“size of data” n* that will be used to discuss time/space complexity?**  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***a)* What is the runtime of the method *p.evaluate(x) (*in terms of *n) ?***

**** Ans)** O(n)*\_\_*   
**b) What is the runtime of the method *sum (p, q)* (in terms of *n)*?**

  
 **Ans)** \_ O(n) + O(n) =\_2\* O(n)\_=>\_O(n)\_\_

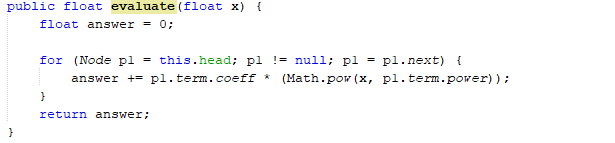
**c) What is the runtime of the method *product (p, q)* (in terms of *n*)?**  
 **Ans)** \_\_O(n2) \_\_

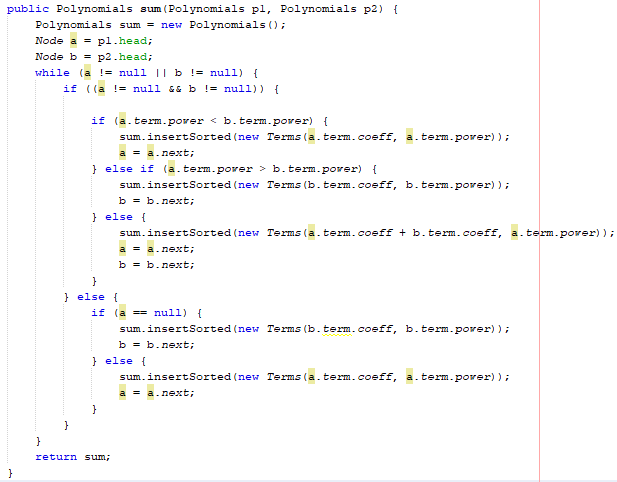
**d) What is the difference between sum and add?**  
**Ans)** Add () Returns the resulting polynomial after adding a polynomial q to the given polynomial whereas Sum () Returns the resulting polynomial after summing two polynomials p and q



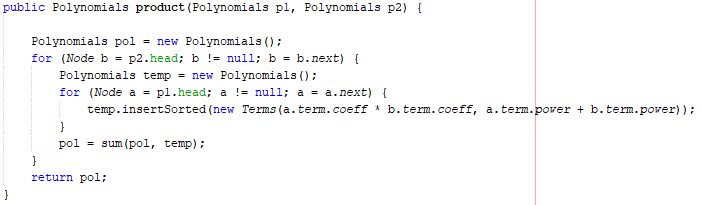
1. **For your linked implementation of the Polynomial class, what would be the best value to use for the *“size of data” n* that will be used to discuss time/space complexity?**  
     
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***a)* What is the runtime of the method *evaluate(x) (*in terms of *n)?***

**Ans)** *O*(n)*\_\_\_\_*



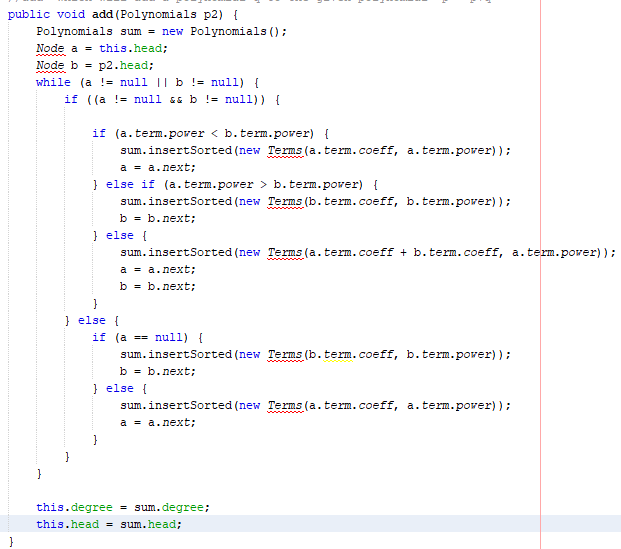
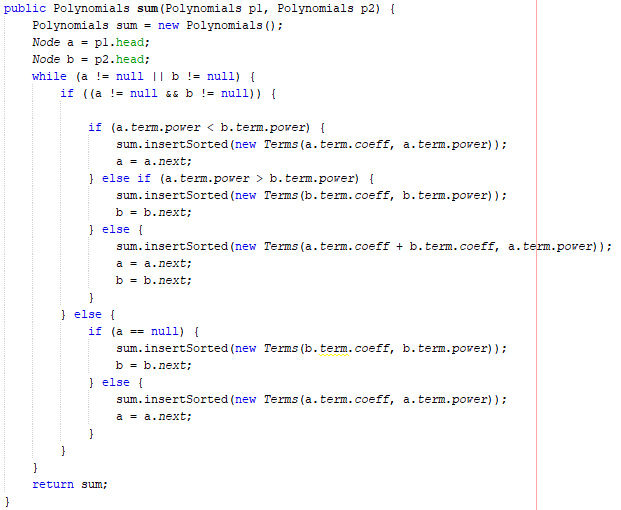
**b) What is the runtime of the method *sum (p, q)* (in terms of *n)*?**   
   
 **Ans)** \_\_O(n)\_\_\_\_

**c) What is the runtime of the method *product (p, q)* (in terms of *n*)?**  
  
 **Ans)** \_\_\_O(n2) \_\_\_

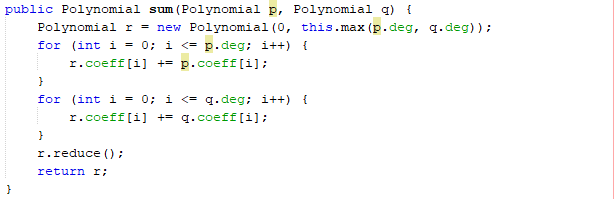


**d) What is the difference between *sum* and *add*?**

**Ans)** Sum () adds 2 polynomials and creates a new polynomial whereas Add () will add a polynomial q to the given polynomial p = p + q.

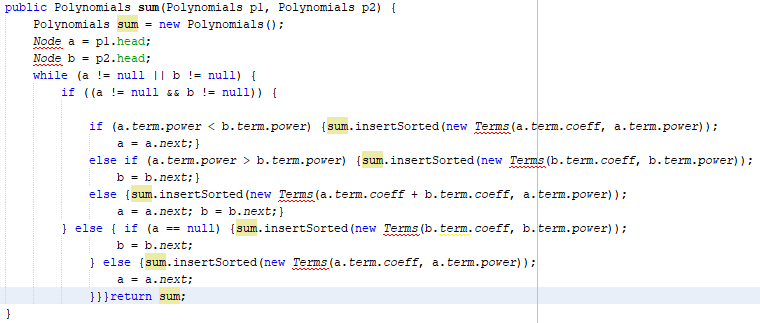


1. **For the following questions, give an example (using *5th* degree polynomials) of:**  
     
   **a) When an array implementation is “better” for the method *sum* and why?**  
   **Ans)** Array implementation will be better if the degree of the polynomial is predefined.



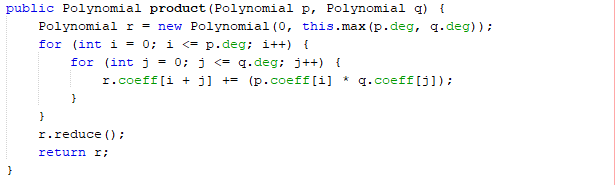
**b) When a linked implementation is “better” for the method *sum* and why?**

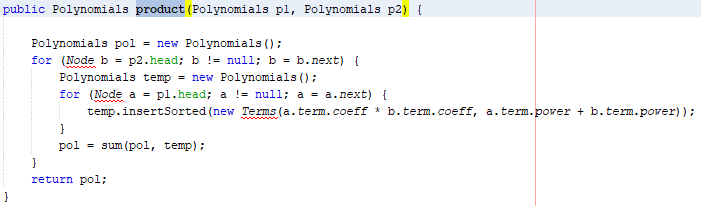
**Ans)** There is a size limit on arrays, which is a limitation. The number of elements in the array must be specified ahead of time. In contrast, the linked list allows you to add as many entries as you like.

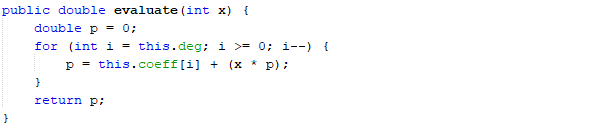
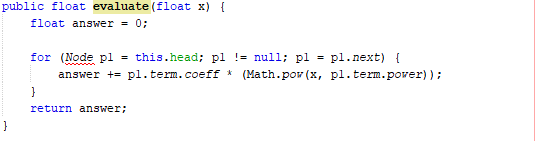


**c) When an array implementation is “better” for the method *product* and why?**

**Ans)** It is better when the terms in polynomials are predefined or already know.

  
**d) When a linked implementation is “better” for the method *product* and why?**  
**Ans)** It is better when we do not have the degrees of the polynomials.

  
  
**e) When an array implementation is “better” for the method *evaluate* and why?**  
**Ans)** Array implementation is better to use for evaluate method when we know the degree and coefficients beforehand.

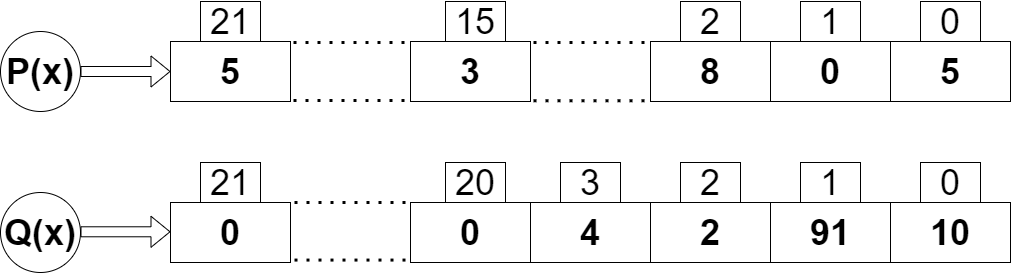
  
**f) When a linked implementation is “better” for the method *evaluate* and why?**  
**Ans)** In evaluate method of polynomials in linked implementation it will perform better when the degree and coefficient are predefined and the coefficient is not a fraction.  


1. **Briefly** **discuss the differences in the space complexity (memory) of the array and linked implementations of the Polynomial class.**

**Ans)** When performing polynomial operations, arrays take up a lot of space. When solving polynomial equations, two parallel data structures are required. Both data structures' indexes advance to perform the desired operation.

P(x)= 5x21+3x15+8x2+5

Q(x)=4x3+2x2+91x+10

Let’s take an example of adding these two.

N values are used to store only four values, as you can see (N being the highest power). There will now be a single pointer that moves from index 21 to 0 and adds elements from both arrays. If we use polynomials with higher powers, there will be a significant amount of memory waste.  
  
The index of any element in an array can be used to access it directly. In the case of a linked list, however, all previous elements must be traversed before any element can be reached. Additionally, better cache locality in arrays (due to contiguous memory allocation) can improve performance significantly. As a result, some operations (like modifying a specific element) are faster in arrays, while others (like inserting/deleting a data element) are faster in linked lists.