# **Week 2**

**1. Read the degree of two polynomials and their coefficients, all integers, from the standard input. The polynomial is of the form ( ) = ∗ + ⋯ + 1 ∗ 1 + , where 0 ≠ 0.**

Step 1: Start.

Step 2: Declare arrays e, f and sum.

Step 3: Declare s, t and size.

Step 4: Set value of e to {12, 8, 7, 14 } and b to {5, 9, 13, 7}.

Step 5: Set value of s to e.length and t to f.length.

Step 6: Display ‘First polynomial is’

Step 7: for (int i<-0; i<s; i++)

Step 7.1: Output e[i]

Step 7.2: if i! <-0

Display x^+i;

Step 7.3: if i! <-x-1

Display +;

Step 8: Display ‘Second Polynomial is’

Step 9: for (int i<-0; i<t; i++)

Step 9.1: Output f[i]

Step 9.2: if i! <-0

Display x^+1;

Step 9.3: if i! <- t-1

Display +;

Step 10: if s>t or s=t

Set value of size to s.

Step 10.1: else

Set value of size to t.

Step 11: Set length of array sum to size.

Step 12: for (int i<-0; i<s; i++)

Step 12.1: sum[i] <-e[i]

Step 13: for (int i<-0; i<t; i++)

Step 13.1: sum[i] <-sum[i] +f[i]

Step 14: Display ‘Sum is’

Step 15: for int (i<=0; i<size; i++)

Step 15.1: Output sum[i]

Step 15.2: if I! <=0

Display x^+i

Step 15:3: if i! <-size-1

Step 16: Stop.

**Write the pseudo code for adding two polynomials.**

Let p and q be the two polynomials represented by the linked list.  
Step1. While p and q are not null, repeat step 2.  
Step2. If powers of the two terms ate equal  
then

Step3.if the terms do not cancel then insert the sum of the terms into the sum Polynomial  
Advance p  
Advance q  
Step4.Else if the power of the first polynomial> power of second  
Then insert the term from first polynomial into sum polynomial  
Advance p  
Step5.Else insert the term from second polynomial into sum polynomial  
Advance q  
Step6.copy the remaining terms from the nonempty polynomial into the  
sum polynomial.

**2. Write the pseudo code and code for a function that determines whether given word is palindrome. What is the time complexity (expressed using BigO notation)?**

Pseudo code:

Step 1: Start

Step 2: Declare variables strChr, length, reverseChr

Step 3: Read strChr, reverseChr

Step4: for( int i=length-1,i>=0;i--)

{

ReverseChr=reverseChr+ str.charAt(i);

}

Step 5: repeat step 4

Step 6:if(strChr equals(reverseChr))

{

Print(“it is palindrome”);

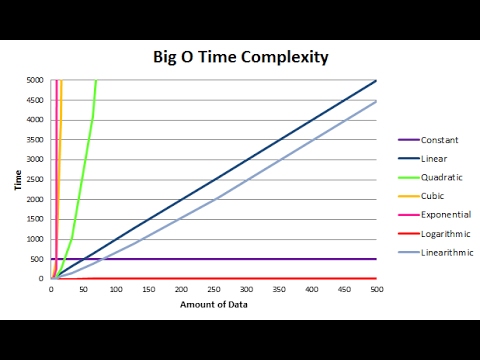
}

Else

Print(“not palindrome”)

}

Step 6: stop



Time complexity is the execution time it takes for your algorithm to solve a problem. To explain it a little further, the time complexity is driven by two things i.e. execution time & the space required by the program.

Big O Notation (a mathematical expression) helps to measure the time complexity by classifying how your program behaves with varying input and taking in different operations.