	PUNE INSTITUTE OF COMPUTER TECHNOLOGY PUNE - 411043	
	Department of Electronics & Telecommunication	
	ASSESMENT YEAR: 2020-2021	CLASS: SE 5
	SUBJECT: DATA STRUCTURES	
EXPT No: 8	LAB Ref: SE/2020-21/	Starting date: 19/11/2020
	Roll No: 22119	Submission date:26/11/2020
Title:	Arithmetic operations on Polynomials	
Prerequisites:	Fundamentals of C	
	Dev C++	
	Linked List	
	Mathematics for Polynomial Addition	
Objectives:	• To learn how to represent polynomial using array and linked list.	
	• Implement polynomial using linked data structure.	
	• Perform and verify mathematical operation on polynomial.	
Theory:		

Linked List: A **linked list** is a linear collection of data elements whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence.

Polynomial: A **polynomial** is defined as an expression which is composed of variables, constants and exponents, that are combined using the mathematical operations such as addition, subtraction, multiplication and division (No division operation by a variable).

Algorithm

Step 1: start

Step 2: While p and q are not null, repeat step 3.

Step 3: If powers of the two terms are equal, and if the terms do not cancel then insert the sum of the terms into the sum Polynomial

Step 4: Advance poly Advance poly1

Else if the power of the first polynomial > power of second, then insert the term from first polynomial into sum polynomial. Advance p

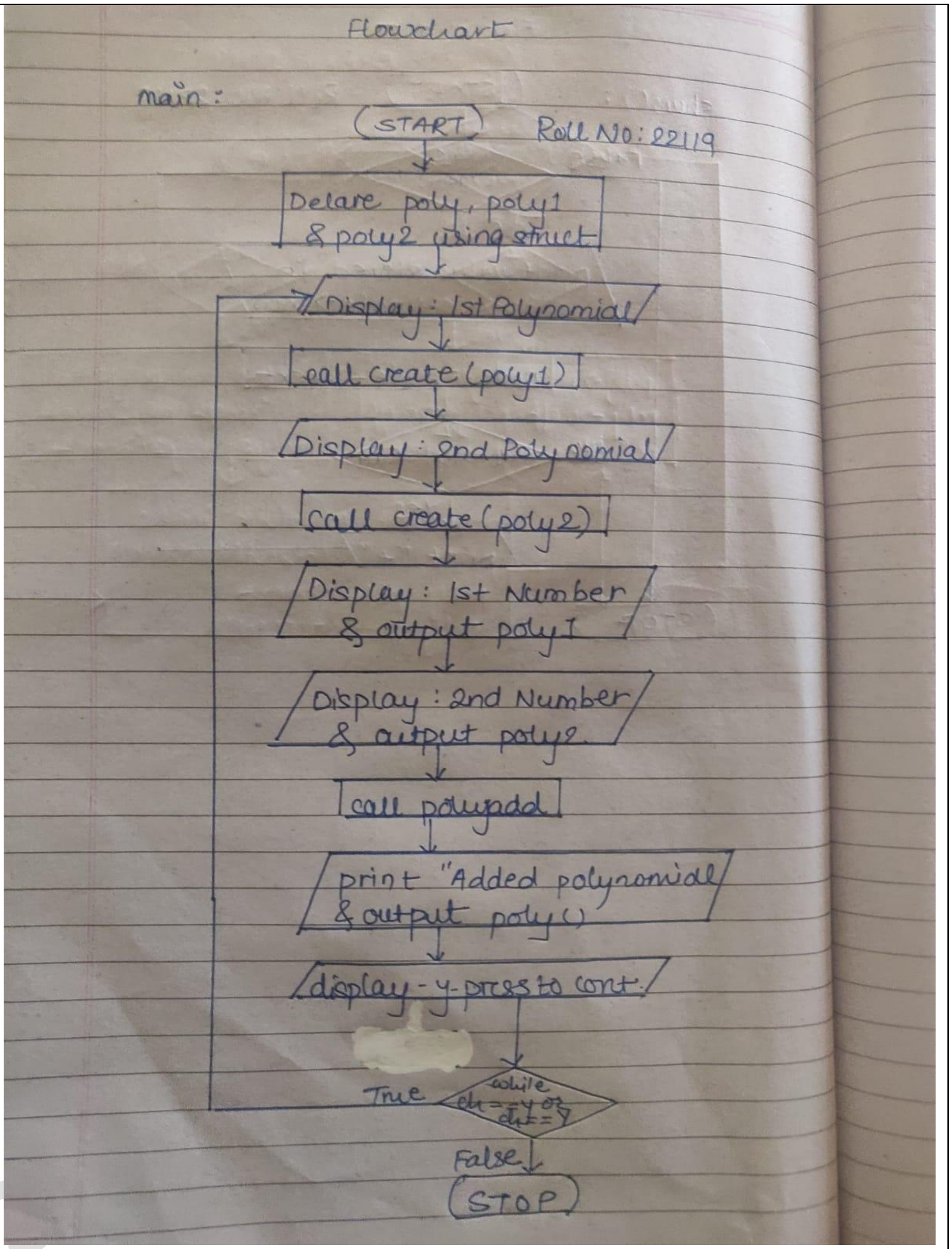
Step 5: Else insert the term from second polynomial into sum polynomial Advance q

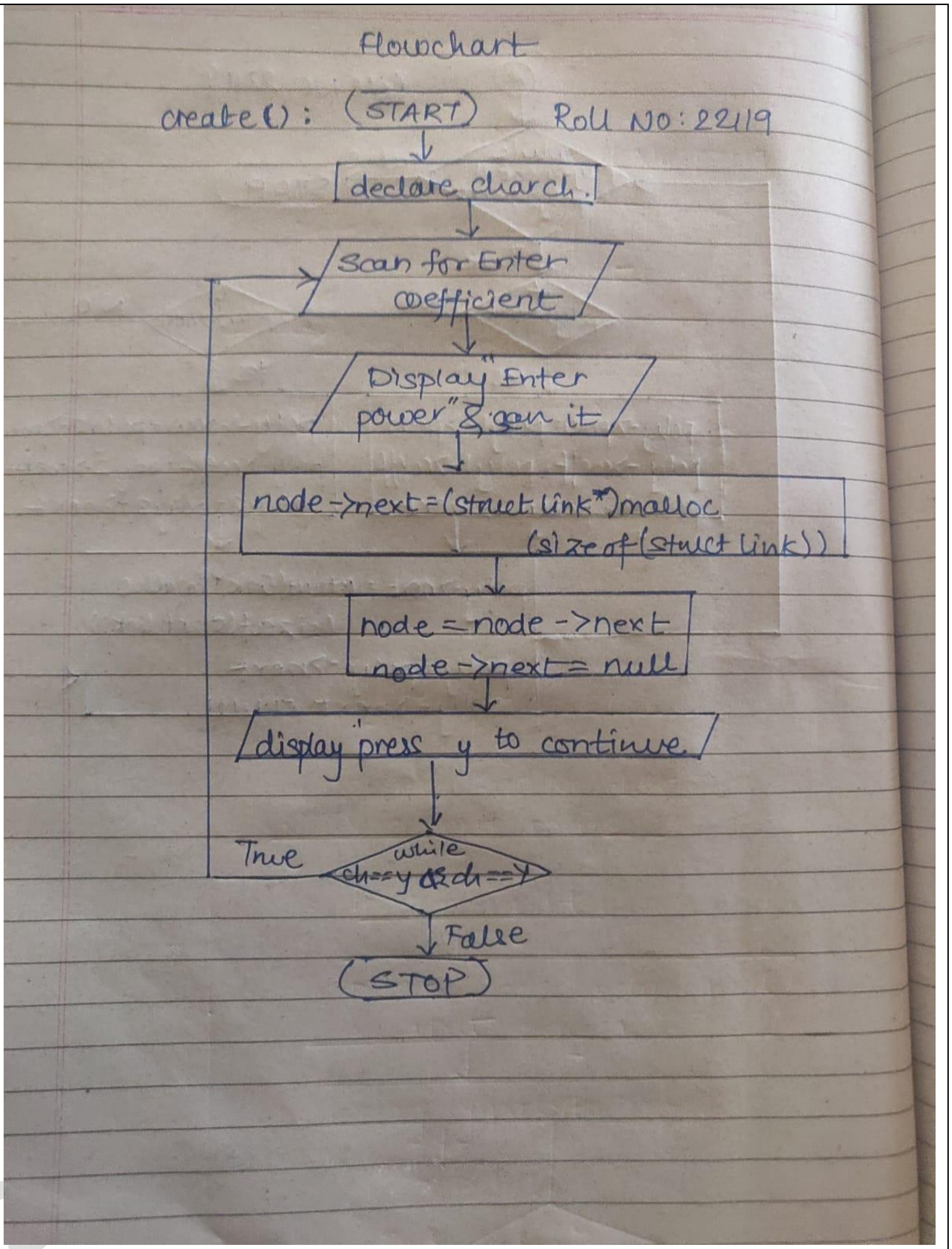
Step 6: Copy the remaining terms from the non-empty polynomial into the sum polynomial.

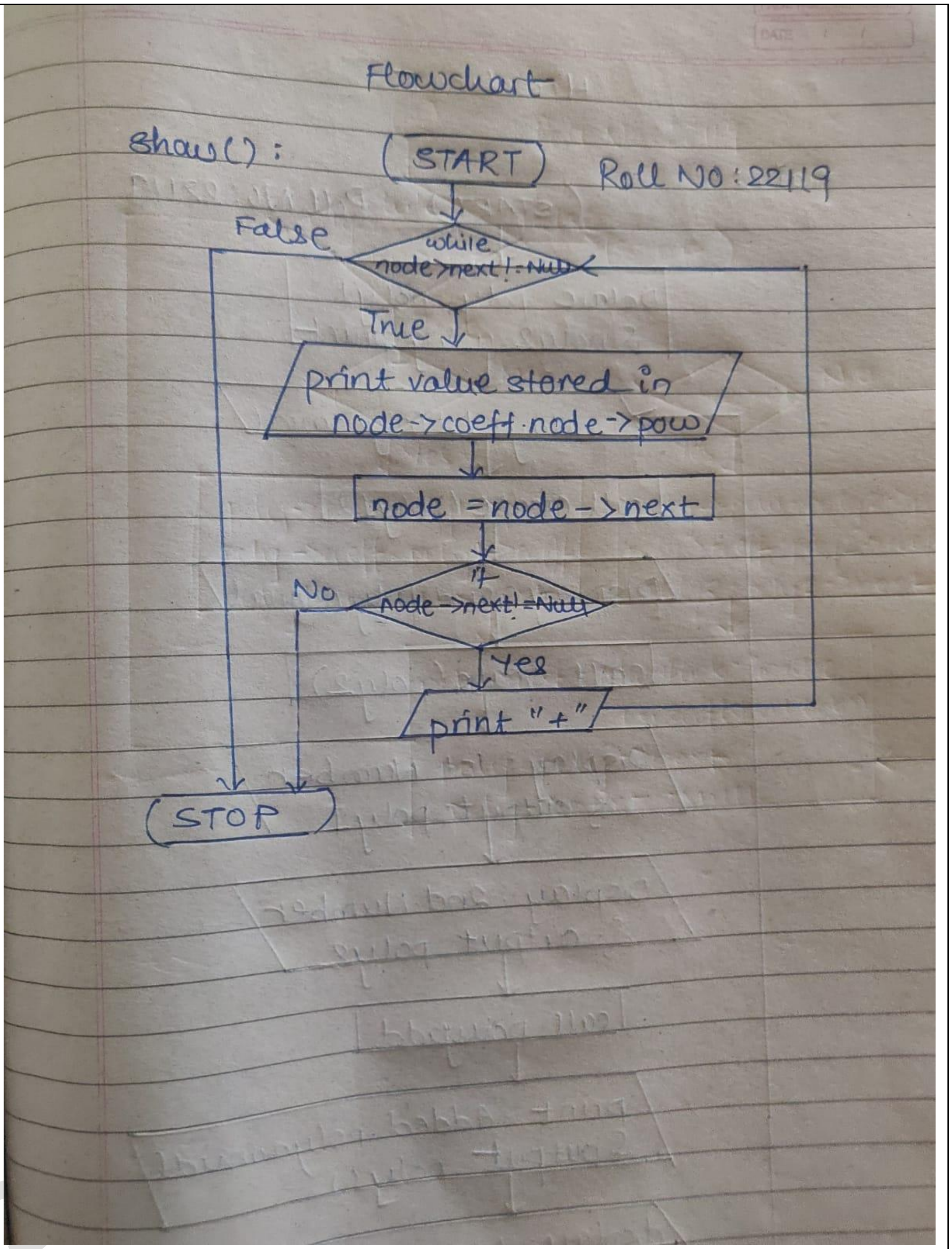
Step 7: The third step of the algorithm is to be processed till the end of the polynomials has not been reached.

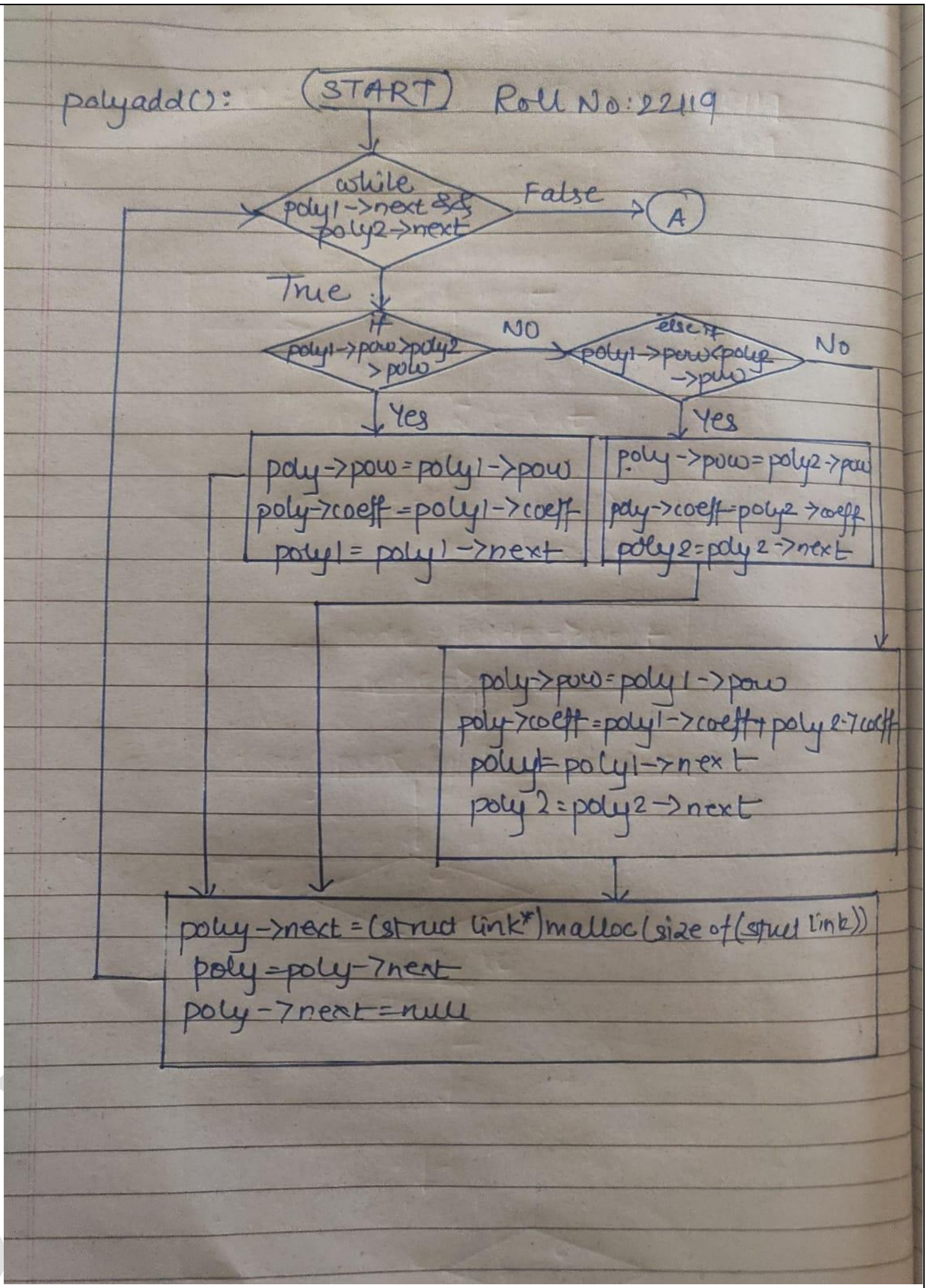
Add_Polynomial:

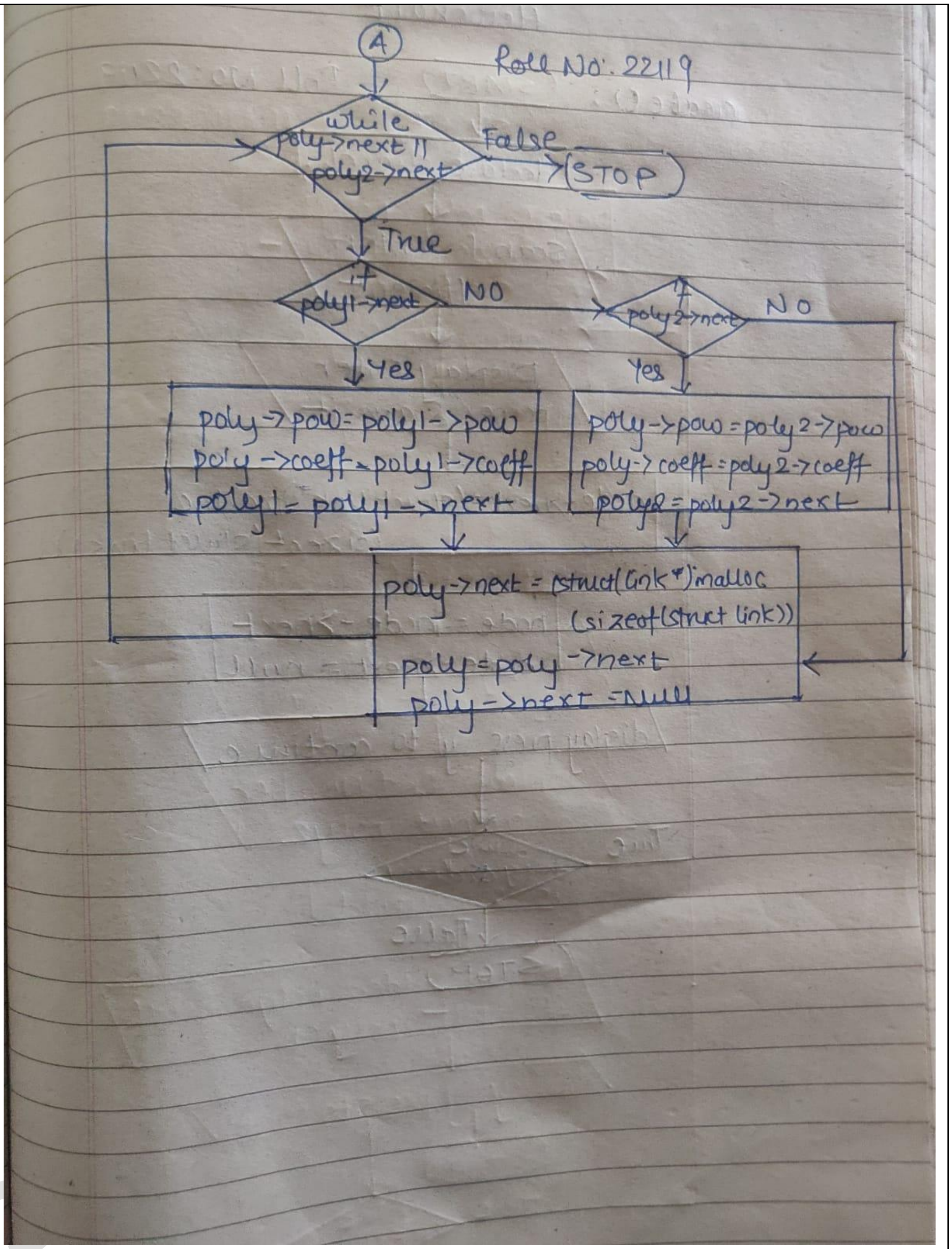
```
Step 1: start
Step 2: set p, q to point to the two first nodes (no headers)
Step 3: initialize a linked list r for a zero polynomial
Step 4: while p!= null and q!=null

        if p.exp > q.exp
          create a node storing p.coeff and p.exp
          insert at the end of list r
          advance p

        else if q.exp > p.exp
          create a node storing q.coeff and q.exp
          insert at the end of list r
          advance q

        else if p.exp == q.exp
          if p.coeff + q.coeff != 0
            create a node storing p.coeff + q.coeff and p.exp
            insert at the end of list r
            advance p , q
step 5: if p != null
         copy the remaining terms of p to end of r
         else go to step 6
step 6: if q != null
        copy the remaining terms of p to end of r
```

Flow-chart	Flowchart main : START Roll No: 22119 Delare poly, poly1 & poly2 using struct Display: 1st Polynomial call create (poly1) Display: 2nd Polynomial call create (poly2) Display: 1st Number & output poly1 Display: 2nd Number & output poly2. call polyadd print "Added polynomial & output poly() display - y - press to cont. while ch=='y' or ch=='Y' True False STOP

	Flowchart create(): START Roll NO: 22119 declare char ch. Scan for Enter coefficient Display "Enter power" & scan it node->next=(struct link*)malloc (sizeof(struct link)) node = node->next node->next = null display press y to continue. while ch=='y' && ch=='Y' True False STOP


Flowchart
show() :
START
Roll NO : 22119
False
while node->next != Null
True
print value stored in node->coeff.node->pow
node = node->next
No
if node->next != Null
Yes
print "+"
STOP


polyadd():
START
Roll No: 22119
while poly1->next && poly2->next
False
A
True
if poly1->pow>poly2->pow
NO
else if poly1->pow<poly2->pow
No
Yes
Yes
poly->pow = poly1->pow
poly->coeff = poly1->coeff
poly1 = poly1->next
poly->pow = poly2->pow
poly->coeff = poly2->coeff
poly2 = poly2->next
poly->pow = poly1->pow
poly->coeff = poly1->coeff + poly2->coeff
poly1 = poly1->next
poly2 = poly2->next
poly->next = (struct link*)malloc(size of(struct link))
poly = poly->next
poly->next = null

	A Roll No: 22119 while poly->next || poly2->next False STOP True if poly1->next NO if poly2->next NO Yes Yes poly->pow=poly1->pow poly->coeff=poly1->coeff poly1=poly1->next poly->pow=poly2->pow poly->coeff=poly2->coeff poly2=poly2->next poly->next=(struct(link*)malloc (sizeof(struct link)) poly=poly->next poly->next=NULL

ERROR	No errors found
REMEDY	none

CONCLUSION:

- Learnt to represent a polynomial using linked list
- Performed mathematical operation ie addition on polynomials using linked list

REFERENCES:

1. Seymour Lipschutz, Data Structure with C, Schaum's Outlines, Tata McGrawHill
2. Yedidyah Langsam – Data structures using C and C++ - PHI Publications (2nd Edition).
3. E Balgurusamy - Programming in ANSI C, Tata McGraw-Hill (Third Edition)

Continuous Assessment			**Assessed By**
RPP (5)	**ARR (5)**	**Total (10)**	**Signature:**
			Date: