1. a) Linear Search.

Code:- #include<stdio.h>

int main(){

    int i,n,key,index=-1;

    printf("Enter size of array: ");

    scanf("%d",&n);

    int a[n];

    printf("Enter array elements: ");

    for(i=0;i<n;i++){

        scanf("%d",&a[i]);

    }

    printf("The array elements are: \n");

    for(i=0;i<n;i++){

        printf("%d\n",a[i]);

    }

    printf("Enter the searching elCodement: ");

    scanf("%d",&key);

    for(i=0;i<n;i++){

        if(key==a[i]){

            index=i;

            break;

        }

    }

    if(index!=-1){

        printf("%d is found %d index",key,index);

    }else{

        printf("%d is not found",key);

    }

}

OUTPUT:- Enter size of array: 5

Enter array elements: 15 54 21 65 9

The array elements are:

15

54

21

65

9

Enter the searching element: 65. 65 is found 3 index.

b) Binary Search.

Code:- #include<stdio.h>

int main(){

    int i,n,key,index=-1;

    printf("Enter size of array: ");

    scanf("%d",&n);

    int a[n];

    printf("Enter array elements: ");

    for(i=0;i<n;i++){

        scanf("%d",&a[i]);

    }

    printf("The array elements are: \n");

    for(i=0;i<n;i++){

        printf("%d\n",a[i]);

    }

    printf("Enter the searching element: ");

    scanf("%d",&key);

    int first=0,last=n-1,mid;

    while(first<=last){

        mid=(first+last)/2;

        if(key==a[mid]){

            index=mid;

            printf("%d is found at %d index",key,index);

            return 0;

        }else if(a[mid]<key){

            first=mid+1;

        }else{

            last=mid-1;

        }

    }

        printf("%d is not found\n",key);

        return 0;

}

OUTPUT:- Enter size of array: 5

Enter array elements: 10 20 30 40 50

The array elements are:10 20 30 40 50

Enter the searching element: 40

40 is found at 3 index

1. Bubble Selection Sort.

Code:-

#include<stdio.h>

void bubblesort(int a[],int n){

    int i,j;

    for(i=0;i<n;i++){

        for(j=i+1;j<n;j++){

            if(a[i]>a[j]){

                int temp=a[i];

                a[i]=a[j];

                a[j]=temp;

            }

        }

    }

}

void selectionsort(int a[],int n){

    int i,j,temp,max;

    for(i=0;i<n;i++){

        max=i;

        for(j=i+1;j<n;j++){

            if(a[j]>a[max]){

                max=j;

            }

        }

            temp=a[max];

            a[max]=a[i];

            a[i]=temp;

    }

}

void main(){

    int i,n;

    printf("Enter the size of the array: ");

    scanf("%d",&n);

    int a[n];

    printf("Enter the %d array elements: \n",n);

    for(i=0;i<n;i++){

        scanf("%d",&a[i]);

    }

    bubblesort(a,n);

    printf("Bubblesort order: \n");

    for(i=0;i<n;i++){

        printf("%d\n",a[i]);

    }

    selectionsort(a,n);

    printf("Selectionsort order: \n");

    for(i=0;i<n;i++){

        printf("%d\n",a[i]);

    }

}

OUTPUT:-

Enter no of array elements: 5

Enter 5 elements: 85 6 0 -45 68

Bubblesort in ascending order: -45 0 65 68 89

Selection sort in descending order: 89 68 65 0 -45

1. Infix to Postfix.

Code:-

#include<stdio.h>

#include<ctype.h>

#define maxsize 50

char stack[maxsize];

int top=-1;

char infix[50],postfix[50];

void push(char item){

    stack[++top]=item;

}

char pop(){

    return stack[top--];

}

int precedence(char symbol){

    if(symbol=='^'){

        return 3;

    }else if(symbol=='\*'||symbol=='/'){

        return 2;

    }else if(symbol=='+'||symbol=='-'){

        return 1;

    }else{

        return 0;

    }

}

void infix\_post(){

    char ch,elem;

    int i=0,k=0;

    stack[++top]='$';

    while((ch=infix[i++])!='\0'){

        if(ch=='('){

            push(ch);

        }else{

            if(isalnum(ch)){

                postfix[k++]=ch;

            }else{

                if(ch==')'){

                    while(stack[top]!='(')

                    postfix[k++]=pop();

                    elem=pop();

                }else{

                    while(precedence(stack[top]>=precedence(ch))){

                        postfix[k++]=pop();

                    }

                    push(ch);

                }

            }

        }

    }

    while(stack[top]!='$')

    postfix[k++]=pop();

    postfix[k]='\0';

}

void main(){

    printf("Enter the infix expression: ");

    scanf("%s",infix);

    push('$');

    infix\_post();

    printf("Postfix expression: %s\n",postfix);

}

OUTPUT:- Enter the infix expression: (a+b\*d)/(e-f)+g

Postfix expression: abd\*+ef-g+/

1. Postfix evaluation.

Code:

#include <stdio.h>

#include <ctype.h>

#define maxsize 100

#define postfixsize 100

int stack[maxsize];

int top = -1;

char postfix[50];

void push(char item) {

    stack[++top] = item;

}

char pop() {

    return stack[top--];

}

void evalpost(char postfix[]) {

    int val, op1, op2, i,ch;

    for (i = 0; postfix[i] != ')'; i++) {

        ch = postfix[i];

        if (isdigit(ch)) {

            push(ch - '0');

        } else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') {

            op2 = pop();

            op1 = pop();

            switch (ch) {

                case '+':

                    val = op1 + op2;

                    break;

                case '-':

                    val = op1 - op2;

                    break;

                case '\*':

                    val = op1 \* op2;

                    break;

                case '/':

                    val = op1 / op2;

                    break;

            }

            push(val);

        }

    }

    printf("\nResult of the expression evaluation: %d\n", pop());

}

void main() {

    int i;

    printf("ASSUMPTIONS: There are only 4 operators (\*, /, -, +) in an expression and operands are single digits only\n");

    printf("\nEnter the postfix expression (press right parenthesis ')' to end the expression):\n");

    for (i = 0; i < postfixsize-1; i++) {

        scanf(" %c", &postfix[i]);

        if (postfix[i] == ')') {

            break;

        }

    }

    evalpost(postfix);

}

OUTPUT:

ASSUMPTIONS: There are only 4 operators (\*, /, -, +) in an expression and operands are single digits only

Enter the postfix expression (press right parenthesis ')' to end the expression): 452\*+5+)

Result of the expression evaluation: 19

1. Dijsktra.

#include<stdio.h>

#include<stdlib.h>

int cost[10][10],n,d[10],p[10],v[10];

void dij(int s){

    int i,count=1,min,u;

    for(i=0;i<n;i++){

        v[i]=0;

        p[i]=s;

        d[i]=cost[s][i];

    }

    v[s]=1;

    while(count<n){

        min=999;

        u=-1;

        for(i=0;i<n;i++){

            if(v[i]==0&&d[i]<min){

                min=d[i];

                u=i;

            }

        }

        v[u]=1;

        for(i=0;i<n;i++){

            if(v[i]==0&&d[u]+cost[u][i]<d[i]){

                d[i]=d[u]+cost[u][i];

                p[i]=u;

            }

        }

        count++;

    }

}

void read\_mat(){

    int i,j;

    for(i=0;i<n;i++)

    for(j=0;j<n;j++)

    scanf("%d",&cost[i][j]);

}

void print\_mat(){

    int i,j;

    for(i=0;i<n;i++){

        for(j=0;j<n;j++){

            printf("%d\t",cost[i][j]);

        }

    printf("\n");

    }

}

void print\_path(int source,int destination){

    int i;

    i=destination;

    while(i!=source){

        printf("%d<--",i);

        i=p[i];

    }

    printf("%d=%d",i,d[destination]);

}

void main(){

    int s,j;

    printf("Enter the no of vertices: ");

    scanf("%d",&n);

    printf("Enter source vertex: ");

    scanf("%d",&s);

    printf("Enter cam: ");

    read\_mat();

    printf("===CST===\n");

    print\_mat();

    dij(s);

    printf("\nThe shortest path and distance are shown below:\n");

    for(j=0;j<n;j++){

        print\_path(s,j);

        printf("\n");

    }

}

OUTPUT:

Enter the no of vertices: 5

Enter source vertex: 0

Enter cam:

0 3 999 7 999

3 0 4 2 999

999 4 0 5 6

7 2 5 0 4

999 999 6 4 0

===CST===

0 3 999 7 999

3 0 4 2 999

999 4 0 5 6

7 2 5 0 4

999 999 6 4 0

The shortest path and distance are shown below:

0=0

1<--0=3

2<--1<--0=7

3<--1<--0=5

4<--3<--1<--0=9

1. Kruskal.

#include<stdio.h>

#include<stdlib.h>

int cost[10][10],n,v1,v2;

int root[10];

void findmin(){

    int min=999,i,j;

    for(i=0;i<n;i++){

        for(j=0;j<n;j++){

            if(cost[i][j]!=0&&cost[i][j]<=min){

                min=cost[i][j];

                v1=i;

                v2=j;

            }

        }

    }

}

void read\_mat(){

    int i,j;

    for(i=0;i<n;i++)

    for(j=0;j<n;j++)

    scanf("%d",&cost[i][j]);

}

void print\_mat(){

    int i,j;

    for(i=0;i<n;i++){

        for(j=0;j<n;j++){

            printf("%d\t",cost[i][j]);

        }

    printf("\n");

    }

}

void dounion(){

    int temp,i;

    temp=root[v2];

    for(i=0;i<n;i++){

        if(root[i]==temp)

            root[i]=root[v1];

    }

}

void kruskal(){

    int i,edgemin,tot=0;

    for(i=0;i<n;i++)

    root[i]=i;

    i=0;

    while(i!=n-1){

        findmin();

        edgemin=cost[v1][v2];

cost[v1][v2]=cost[v2][v1]=0;

        if(root[v1]!=root[v2]){

            printf("\n%d--->%d=%d\n",v1,v2,edgemin);

            dounion();

            tot+=edgemin;

            i++;

        }

    }

    printf("\nMincost=%d",tot);

}

void main(){

    printf("Enter the no of vertices: ");

    scanf("%d",&n);

    printf("\nEnter cam");

    read\_mat();

    printf("===CAM===\n");

    print\_mat();

    kruskal();

}

OUTPUT: Enter the no of vertices: 6

Enter cam0 3 999 999 6 5

3 0 1 999 999 4

999 1 0 6 999 4

999 999 6 0 8 5

6 999 999 8 0 2

5 4 4 5 2 0

===CAM===

0 3 999 999 6 5

3 0 1 999 999 4

999 1 0 6 999 4

999 999 6 0 8 5

6 999 999 8 0 2

5 4 4 5 2 0

2--->1=1

5--->4=2

1--->0=3

5--->2=4 5--->3=5 Mincost=15

1. LinkedList.

Code:

#include<stdio.h>

#include<stdlib.h>

struct linkedlist{

    int data;

    struct linkedlist \*ptr;

};

typedef struct linkedlist \*node;

node head=NULL;

node createnode(){

    node temp;

    temp=(node)malloc(sizeof(struct linkedlist));

    temp->ptr=NULL;

    return temp;

}

node insert\_end(int ele){

    node temp;

    temp=createnode();

    temp->data=ele;

    if(head==NULL){

        head=temp;

    }else{

        temp->ptr=head;

        head=temp;

    }

    return head;

}

node insert\_begin(int ele){

    node temp,p;

    temp=createnode();

    temp->data=ele;

    if(head==NULL){

        head=temp;

    }else{

        p=head;

        while(p->ptr!=NULL){

            p=p->ptr;

        }

        p->ptr=temp;

    }

    return head;

}

node delete\_begin(){

    node temp;

    temp=head;

    if(head==NULL){

        printf("Empty list");

    }else if(head->ptr==NULL)

        head=NULL;

        else

        head=head->ptr;

        free(temp);

    return head;

}

node delete\_end(){

    node temp,prev;

    temp=head;

    if(head==NULL)

        printf("\nEmpty list");

    else if(head->ptr==NULL)

        head=NULL;

    else{

        while(temp->ptr!=NULL){

            prev=temp;

            temp=temp->ptr;

        }

        prev->ptr=NULL;

    }

    free(temp);

    return head;

}

node delete\_specific(int ele){

    node temp,prev;

    temp=head;

    if(head==NULL)

        printf("\nEmpty list");

    else if(head->data==ele){

        if(head->ptr!=NULL)

            head=head->ptr;

        else

            head=NULL;

        }else{

            while(temp->data!=ele && temp->ptr!=NULL){

                prev=temp;

                temp=temp->ptr;

            }

            if(temp->ptr==NULL && temp->data!=ele)

                printf("Specified item not found");

        else if(temp->ptr!=NULL){

            prev->ptr=temp->ptr;

        }else{

            prev->ptr=NULL;

        }

        free(temp);

        return head;

}

}

void display(){

    node temp;

    temp=head;

    if(head==NULL){

        printf("Empty list.");

        return;

    }

    printf("\nelements of the list are: \n");

    while(temp->ptr!=NULL){

        printf("%d\t",temp->data);

        temp=temp->ptr;

    }

    printf("%d\t",temp->data);

}

int main(){

    int ele,opt=1,ch;

    printf("\nSingly linked list\n");

    while(opt){

    printf("\n1.INSERT END\n2.DISPLAY\n3.INSERT BEGIN\n4.DELETE BEGIN\n5.DELETE END\n6.DELETE SPECIFIC\n");

    printf("Enter your choice: ");

    scanf("%d",&ch);

    switch(ch){

        case 1:

        printf("\nEnter the value to be inserted: ");

        scanf("%d",&ele);

        insert\_begin(ele);

        break;

    case 2:

        display();

        break;

    case 3:

        printf("\nEnter the value to be inserted: ");

        scanf("%d",&ele);

        insert\_end(ele);

        break;

    case 4:

        delete\_begin();

        break;

    case 5:

        delete\_end();

        break;

    case 6:

        printf("\nEnter the key value: ");

        scanf("%d",&ele);

        delete\_specific(ele);

        break;

    default:

        printf("Wrong choice!!");

        return 0;

    }

    printf("\nDo you want to continue(0/1)?");

    scanf("%d",&opt);

}

return 0;

}

OUTPUT:

Singly linked list

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 1

Enter the value to be inserted: 10

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 2

elements of the list are: 10

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 1

Enter the value to be inserted: 23

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 2

elements of the list are: 10 23

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 3

Enter the value to be inserted: 31

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 2

elements of the list are: 31 10 23

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 6

Enter the key value: 10

Do you want to continue(0/1)?1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 2

elements of the list are: 31 23

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 5

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 2

elements of the list are: 31

Do you want to continue(0/1)?1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 4

Do you want to continue(0/1)? 1

1.INSERT END

2.DISPLAY

3.INSERT BEGIN

4.DELETE BEGIN

5.DELETE END

6.DELETE SPECIFIC

Enter your choice: 2

Empty list.

Do you want to continue(0/1)?0

PS C:\C Programs>

1. Queue.

Code:

#include <stdio.h>

#include <stdlib.h>

#define ms 5

int q[ms];

int f=-1;

int r=-1;

void insert(int ele) {

    if(r==ms-1){

        printf("Queue is full!\n");

        return;

    }

    if(f==-1){

        f=0;

    }

    q[++r]=ele;

}

void display(){

    int i;

    if (f==-1){

        printf("\nQueue is empty!\n");

        return;

    }

    printf("\nQueue elements available are: ");

    for(i=f;i<=r;i++){

        printf("%d ",q[i]);

    }

    printf("\n");

}

void delete(){

    if (f==-1) {

        printf("Queue is empty!\n");

        return;

    }

    printf("\nDeleted element from the queue is: %d\n", q[f++]);

    if (f>r){

        f=r=-1;

    }

}

int main(){

    int ch, ele;

    while(1){

        printf("\n1.INSERT\n2.DELETE\n3.DISPLAY\n4.EXIT\n");

        printf("Enter your choice: ");

        scanf("%d", &ch);

        switch(ch){

            case 1:

                printf("\nEnter element to insert: ");

                scanf("%d", &ele);

                insert(ele);

                break;

            case 2:

                delete();

                break;

            case 3:

                display();

                break;

            case 4:

                printf("\nEXITING PROGRAM!\n");

                exit(0);

            default:

                printf("INVALID CHOICE!\n");

        }

    }

    return 0;

}

OUTPUT:

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 1

Enter element to insert: 10

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 1

Enter element to insert: 20

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 1

Enter element to insert: 30

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 1

Enter element to insert: 40

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 1

Enter element to insert: 50

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 3

Queue elements available are: 10 20 30 40 50

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 1

Enter element to insert: 60

Queue is full!

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 2

Deleted element from the queue is: 10

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 3

Queue elements available are: 20 30 40 50

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 2

Deleted element from the queue is: 20

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 3

Queue elements available are: 30 40 50

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 2

Deleted element from the queue is: 30

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 3

Queue elements available are: 40 50

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 2

Deleted element from the queue is: 40

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 3

Queue elements available are: 50

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 2

Deleted element from the queue is: 50

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 3

Queue is empty!

1.INSERT

2.DELETE

3.DISPLAY

4.EXIT

Enter your choice: 4

EXITING PROGRAM!

PS C:\C Programs>

1. Polynomial.

Code:

#include <stdio.h>

#include <stdlib.h>

struct node{

 int coef;

 int expo;

 struct node \*link;

};

struct node \*attach(int coef1,int expo1,struct node \*start)

{

 struct node \*temp,\*dstart;

 temp=(struct node \*)malloc(sizeof(struct node));

 temp->coef=coef1;

 temp->expo=expo1;

 dstart=start->link;

 while(dstart->link!=start)

 dstart=dstart->link;

 dstart->link=temp;

 temp->link=start;

 return start;

}

struct node \*readpoly(struct node \*start){

 char ch='Y';

 int coef,expo;

 while(ch=='Y') {

 printf("Enter Coefficient and Exponent\n");

 scanf("%d%d",&coef,&expo);

 start=attach(coef,expo,start);

 printf("Do you wish to enter any more enter(Y/N)\n");

 ch=getchar();

scanf("%c",&ch);

 }

 return start;

}

int compare(int a,int b)

{

 if(a<b)

 return -1;

 if(a==b)

 return 0;

 if(a>b)

 return 1;

}

struct node \*polyadd(struct node \*startA,struct node \*startB,struct node \*startC)

{

 int sum;

 struct node \*dstartA,\*dstartB;

 dstartA=startA->link;

 dstartB=startB->link;

 while(dstartA!=startA && dstartB!=startB) {

 switch(compare(dstartA->expo,dstartB->expo)) {

 case 1:

 startC=attach(dstartA->coef,dstartA->expo,startC);

 dstartA=dstartA->link;

 break;

 case 0:

 sum=dstartA->coef+dstartB->coef;

 startC=attach(sum,dstartA->expo,startC);

 dstartA=dstartA->link;

 dstartB=dstartB->link;

 break;

 case -1:

 startC=attach(dstartB->coef,dstartB->expo,startC);

 dstartB=dstartB->link;

 break;

 }

 }

 while(dstartA!=startA) {

 startC=attach(dstartA->coef,dstartA->expo,startC);

 dstartA=dstartA->link;

 }

 while(dstartB!=startB) {

 startC=attach(dstartB->coef,dstartB->expo,startC);

 dstartB=dstartB->link;

 }

return startC;

}

int main()

{

 struct node \*startA,\*startB,\*startC,\*dstartA,\*dstartB,\*dstartC;

 startA=(struct node \*)malloc(sizeof(struct node));

 startB=(struct node \*)malloc(sizeof(struct node));

 startC=(struct node \*)malloc(sizeof(struct node));

 startA->link=startA;

 startB->link=startB;

 startC->link=startC;

 printf("Reading Polynomial A\n");

 startA=readpoly(startA);

 printf("Reading Polynomial B\n");

 startB=readpoly(startB);

 dstartA=startA->link;

 dstartB=startB->link;

 printf("Polynomial A\n");

 while(dstartA!=startA) {

 printf("%d^%d",dstartA->coef,dstartA->expo);

 dstartA=dstartA->link;

 if(dstartA!=startA)

 printf("+");

 }

 printf("\nPolynomial B\n");

 while(dstartB!=startB) {

 printf("%d^%d",dstartB->coef,dstartB->expo);

 dstartB=dstartB->link;

 if(dstartB!=startB)

 printf("+");

 }

 startC=polyadd(startA,startB,startC);

 dstartC=startC->link;

 printf("\nPolynomial C\n");

 while(dstartC!=startC) {

 printf("%d^%d",dstartC->coef,dstartC->expo);

 dstartC=dstartC->link;

 if(dstartC!=startC)

 printf("+");

 }

 return 0;

}

OUTPUT:

Reading Polynomial A

Enter Coefficient and Exponent

5 2

Do you wish to enter any more enter(Y/N)

Y

Enter Coefficient and Exponent

6 1

Do you wish to enter any more enter(Y/N)

Y

Enter Coefficient and Exponent

1 0

Do you wish to enter any more enter(Y/N)

N

Reading Polynomial B

Enter Coefficient and Exponent

6 1

Do you wish to enter any more enter(Y/N)

Y

Enter Coefficient and Exponent

8 0

Do you wish to enter any more enter(Y/N)

N

Polynomial A

5^2+6^1+1^0

Polynomial B

6^1+8^0

Polynomial C

5^2+12^1+9^0