**Exercise 2: Polynomial manipulation using Linked List**

**polyADT.h:**

#include<stdio.h>

struct poly

{

int coeff;

int exp;

struct poly \*next;

};

struct poly \* insertEnd(struct poly \*p,int c,int e);

void disp(struct poly \*p);

struct poly \*mul(struct poly \*p1,struct poly \*p2);

struct poly \*add(struct poly \*p1,struct poly \*p2);

struct poly \* simp(struct poly \*p);

void polyDegree(struct poly \*p);

int polyEvaluate(struct poly \*p,int x);

**polyimpl.h:**

#include "polyADT.h"

#include<stdlib.h>

#include<math.h>

#include<limits.h>

struct poly \* insertEnd(struct poly \*p,int c,int e)

{

struct poly \*ptr=p;

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

temp->exp=e,temp->coeff=c,temp->next=NULL;

if(ptr==NULL)

{

p=temp;

}

else

{

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

temp->next=NULL;

}

return p;

}

void disp(struct poly \*p)

{

struct poly \*ptr;

ptr=(struct poly \*)malloc(sizeof(struct poly));

ptr=p;

while(ptr!=NULL)

{

ptr->coeff?printf("%dx^%d + ",ptr->coeff,ptr->exp):printf("0 + ");

ptr=ptr->next;

}

printf("\b\b \b");

polyDegree(p);

}

struct poly \*mul(struct poly \*p1,struct poly \*p2)

{

struct poly \*ptr1=p1,\*ptr2=p2;

struct poly \*p3=NULL;

while(ptr1!=NULL)

{

while(ptr2!=NULL)

{

p3=insertEnd(p3,ptr1->coeff\*ptr2->coeff,ptr1->exp+ptr2->exp);

ptr2=ptr2->next;

}

ptr2=p2;

ptr1=ptr1->next;

}

return p3;

}

struct poly \*add(struct poly \*p1,struct poly \*p2)

{

struct poly \*ptr1=p1,\*ptr2=p2;

struct poly \*R=NULL;

while(ptr1 && ptr2)

{

if(ptr1->exp>ptr2->exp)

{

R=insertEnd(R,ptr1->coeff,ptr1->exp);

ptr1=ptr1->next;

}

else if(ptr1->exp<ptr2->exp)

{

R=insertEnd(R,ptr2->coeff,ptr2->exp);

ptr2=ptr2->next;

}

else

{

R=insertEnd(R,ptr1->coeff+ptr2->coeff,ptr2->exp);

ptr1=ptr1->next;

ptr2=ptr2->next;

}

}

while(ptr1||ptr2)

{

if(ptr1)

{

R=insertEnd(R,ptr1->coeff,ptr1->exp);

ptr1=ptr1->next;

}

if(ptr2)

{

R=insertEnd(R,ptr2->coeff,ptr2->exp);

ptr2=ptr2->next;

}

}

return R;

}

struct poly \* simp(struct poly \*p)

{

struct poly \*ptr1=p,\*ptr2=NULL,\*temp=NULL;

while (ptr1 && ptr1->next)

{

ptr2=ptr1;

while(ptr2->next)

{

if (ptr1->exp == ptr2->next->exp)

{

ptr1->coeff=ptr1->coeff+ptr2->next->coeff;

temp= ptr2->next;

ptr2->next = ptr2->next->next;

free(temp);

}

else

ptr2=ptr2->next;

}

ptr1=ptr1->next;

}

return p;

}

void polyDegree(struct poly \*p)

{

int max=INT\_MIN;

struct poly \*ptr=p;

while(ptr)

{

if(max<ptr->exp && ptr->coeff!=0)

{

max=ptr->exp;

}

ptr=ptr->next;

}

if(max!=INT\_MIN)

printf(" , Degree: %d\n",max);

else

printf(" , Degree: Undefined\n");

}

int polyEvaluate(struct poly \*p,int x)

{

int total=0;

struct poly \*ptr=p;

while(ptr)

{

total+=(ptr->coeff\*pow(x,ptr->exp));

ptr=ptr->next;

}

return total;

}

**polyappl.c:**

#include<stdio.h>

#include "polyimpl.h"

int main()

{

struct poly \*p= NULL,\*q=NULL;

struct poly \*r=NULL;

int c,e,n1,n2,ch,x,k=0,ca=0;

char X;

do

{

if(k)

{

printf("\n--------------------------------------\np(x): ");

disp(p);

printf("q(x): ");

disp(q);

if(ca!=0)

{

printf("r(x): ");

disp(r);

}

printf("\nMenu:\n1.Add 2 polynomials\n2.Multiply 2 polynomials\n3.Evaluate polynomial\n4.Exit%s\n\nEnter your choice: ",k!=0?"\n5.Create 2 new poly":"");

scanf("%d",&ch);

}

else

ch=5;

switch(ch)

{

case 1: r=add(p,q);

printf("\nAddition:\nResultant r(x): ");

disp(r);

ca=1;

break;

case 2: r=mul(p,q);

printf("\nMultiplication:\nResultant r(x): ");

disp(r);

r=simp(r);

printf("\nSimplified Resultant r(x): ");

disp(r);

ca=1;

break;

case 3: printf("\nEvaluate polynomial:");

printf("\nChoose which poly to evaluate(p,q%s): ",ca!=0?",r":"");

scanf(" %c",&X);

printf("\nEnter value for x: ");

scanf("%d",&x);

int value;

if(X=='p')

value=polyEvaluate(p,x);

else if(X=='q')

value=polyEvaluate(q,x);

else if(X=='r')

value=polyEvaluate(r,x);

else

{

printf("\nInvalid Input!\n");

break;

}

printf("\n%c(%d): %d\n",X,x,value);

break;

case 4: return 0;

case 5: printf("\nHow many terms in poly 1: ");

scanf("%d",&n1);

p=NULL;q=NULL;

for(int i=0;i<n1;i++)

{

printf("Enter coeff and exp: ");

scanf("%d%d",&c,&e);

p=insertEnd(p,c,e);

}

printf("\nEntered p(x): ");

disp(p);

printf("\nSimplified p(x): ");

p=simp(p);

disp(p);

printf("\nHow many terms in poly 2: ");

scanf("%d",&n2);

for(int i=0;i<n2;i++)

{

printf("Enter coeff and exp: ");

scanf("%d%d",&c,&e);

q=insertEnd(q,c,e);

}

printf("\nEntered q(x): ");

disp(q);

printf("\nSimplified q(x): ");

q=simp(q);

disp(q);

ca=0;

break;

default: printf("\nInvalid Input!\n");

}

k++;

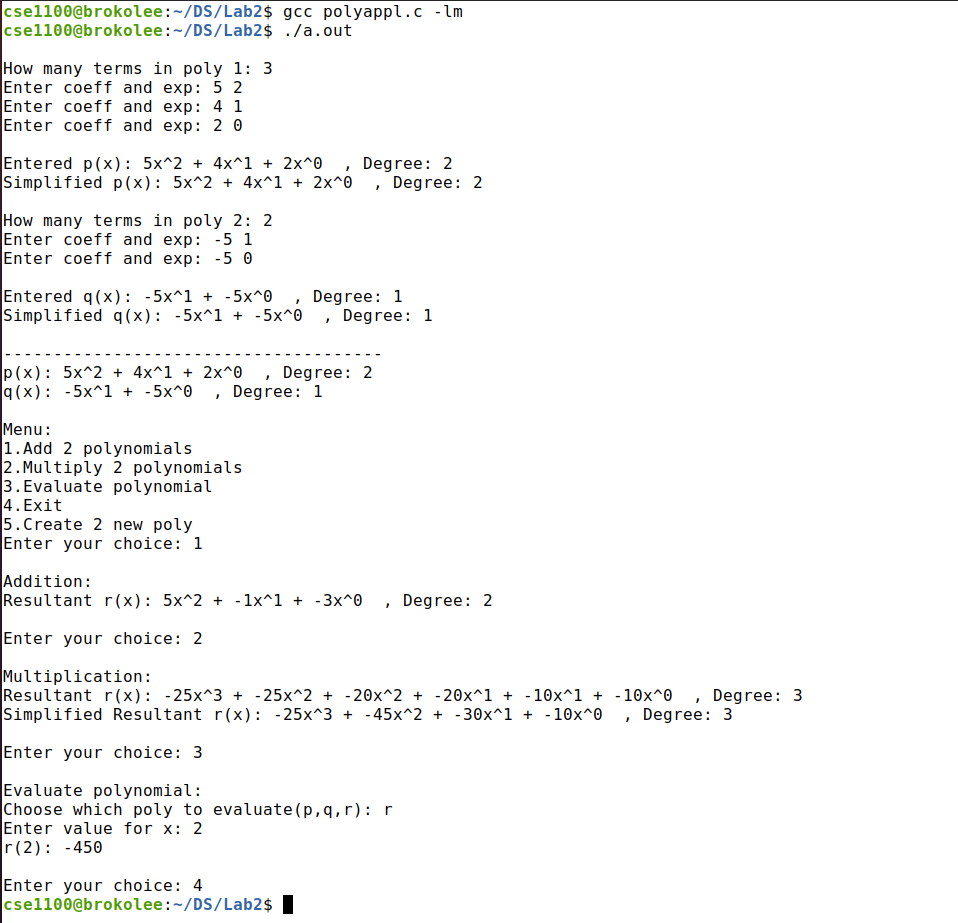
}while(ch!=4);

return 0;

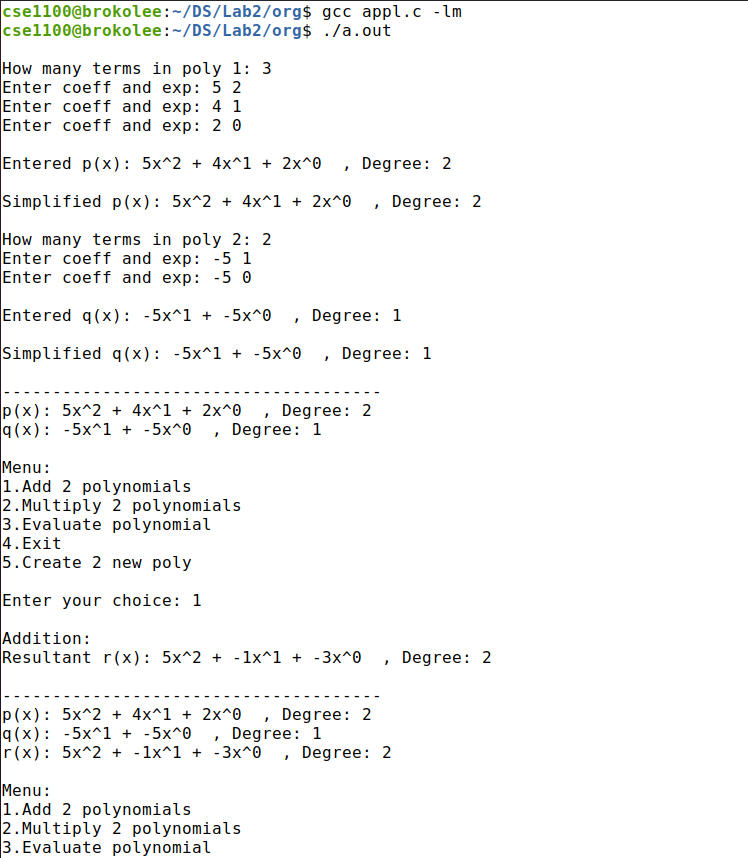
}

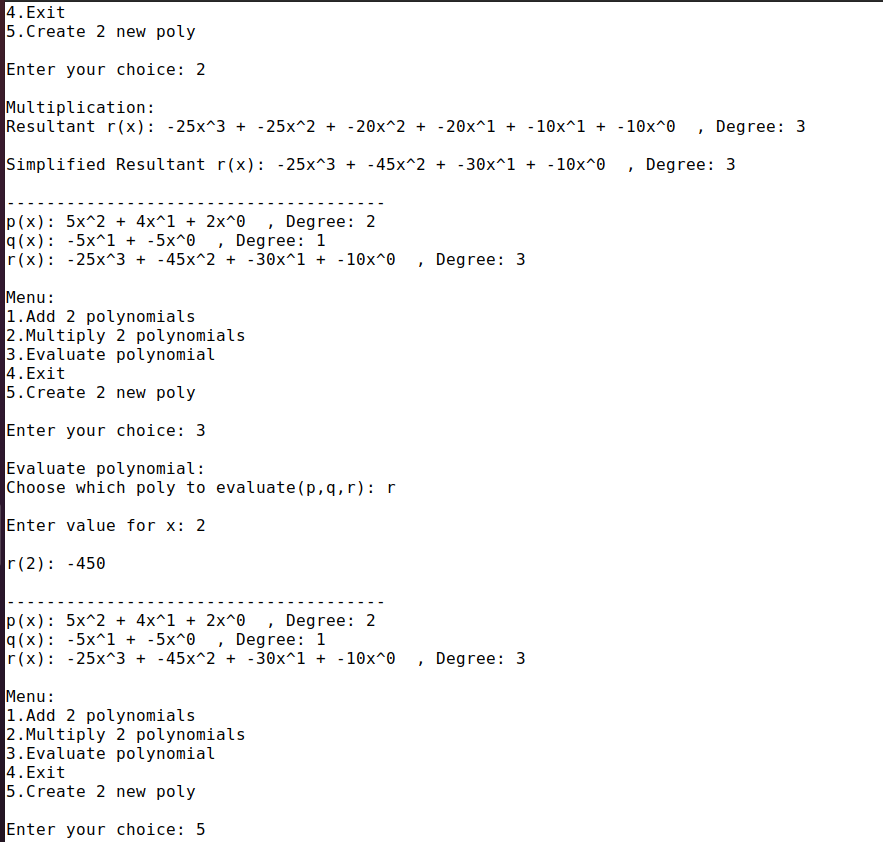
**Sample I/O:**

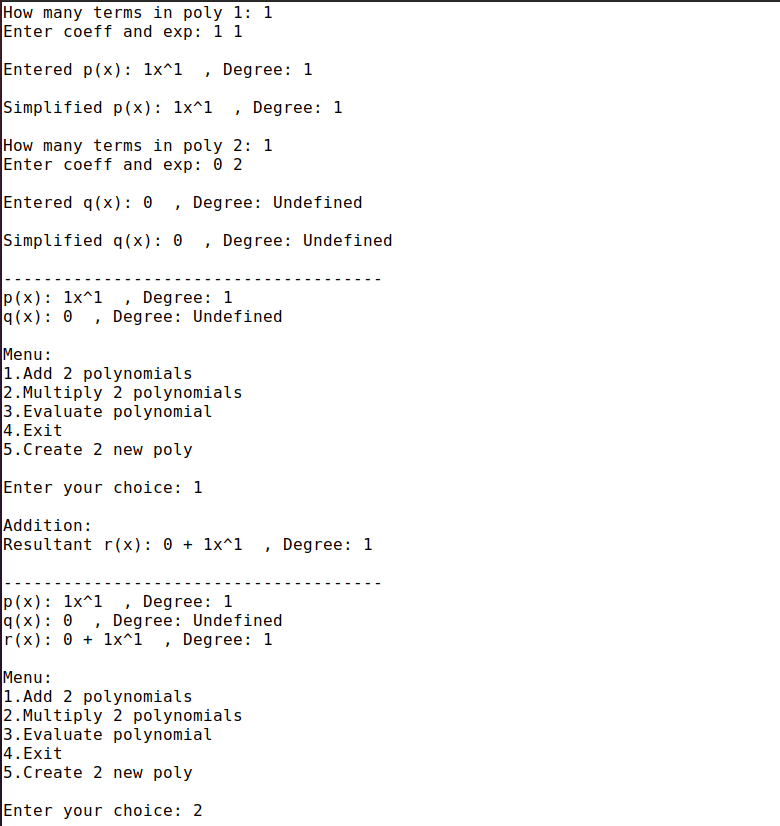
**(new one):**

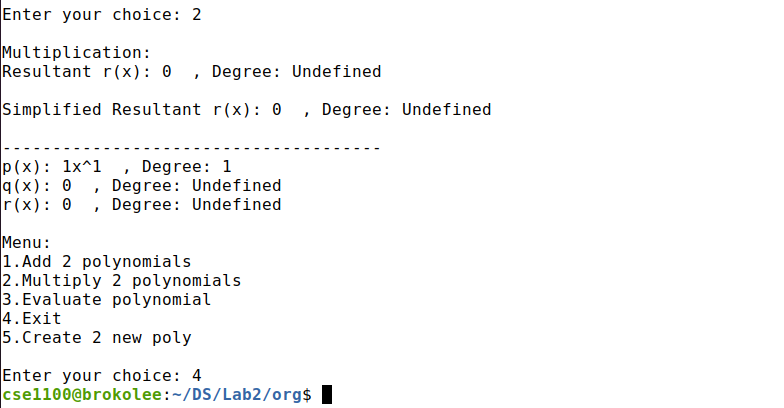


**(old one):**







****