**PROGRAM CODE:**

#include <stdio.h>

#include <malloc.h>

#include "polyadt.h"

void main()

{

int option;

polynomial poly1,poly2,sum\_poly,pro\_poly;

do

{

printf("\n\*\*\*\*\*\*\* MAIN MENU \*\*\*\*\*\*\*");

printf("\n 1. Enter the first polynomial");

printf("\n 2. Enter the second polynomial");

printf("\n 3. Add the polynomials");

printf("\n 4. Multiply the polynomials");

printf("\n 5. Display the sum of the polynomials");

printf("\n 6. Display the product of the polynomials: ");

printf("\n 7. Exit");

printf("\nEnter your option : ");

scanf("%d", &option);

switch(option)

{

case 1: poly1 = create\_poly();

break;

case 2: poly2 = create\_poly();

break;

case 3: sum\_poly = add\_poly(poly1, poly2);

break;

case 4: pro\_poly = mul\_poly(poly1, poly2);

break;

case 5: printf("\nThe sum of the two polynomials is: ");

display\_poly(sum\_poly);

break;

case 6: printf("\nThe product of the two polynomials is: ");

display\_poly(pro\_poly);

break;

case 7:

break;

default:

printf("\nInvalid choice\n");

}

printf("\n\nPress 1 to go back to menu\nPress 2 to exit\n");

scanf("%d",&option);

}while(option==1);

}

**polyadt header file:**

typedef struct poly \*ptrtopoly;

typedef ptrtopoly polynomial;

struct poly

{

float coeff;

int exp;

struct poly \*next;

};

polynomial Createhead()

{

ptrtopoly head;

head=(polynomial)malloc(sizeof(struct poly));

if(head==NULL)

printf("\nPolynomial creation error");

else

head->next=NULL;

return head;

}

polynomial Insert\_term(polynomial start, float c, int e)

{

polynomial ptr,new\_polynomial;

new\_polynomial = (polynomial)malloc(sizeof(struct poly));

new\_polynomial -> exp = e;

new\_polynomial -> coeff = c;

new\_polynomial -> next = NULL;

if(start->next == NULL)

start->next = new\_polynomial;

else

{ ptr = start->next;

while(ptr->next != NULL)

ptr = ptr -> next;

ptr -> next = new\_polynomial;

}

return start;

}

void display\_poly(polynomial start)

{ polynomial ptr = start->next;

if(ptr==NULL)

printf("0 ");

while(ptr!= NULL)

{

if(ptr->exp==0)

printf("%.1f ", ptr -> coeff);

else if (ptr->exp==1)

printf("%.1f x ", ptr -> coeff);

else

printf("%.1f x^%d ", ptr -> coeff, ptr -> exp);

if (ptr->next!=NULL)

if (ptr->next->coeff>0)

printf("+ ");

ptr = ptr -> next;

}

}

polynomial create\_poly()

{ int e; float c;

polynomial poly1=Createhead();

polynomial ptr,new\_polynomial= NULL;

printf("\nEnter 0 as the coefficient to stop\n");

printf("\nEnter the coefficient : ");

scanf("%f", &c);

if(c!=0)

{ printf("Enter the exponent : ");

scanf("%d", &e);

}

while(c != 0)

{

new\_polynomial = (polynomial)malloc(sizeof(struct poly));

new\_polynomial -> coeff = c;

new\_polynomial -> exp = e;

new\_polynomial -> next = NULL;

if(poly1->next==NULL)

poly1->next=new\_polynomial;

else

{

ptr = poly1->next;

while(ptr -> next != NULL)

ptr = ptr -> next;

ptr -> next = new\_polynomial;

}

printf("\nEnter its coefficient : ");

scanf("%f", &c);

if(c!=0)

{ printf("Enter the exponent : ");

scanf("%d", &e);

}

}

printf("\nThe polynomial is: ");

display\_poly(poly1);

return poly1;

}

polynomial add\_poly(polynomial poly1, polynomial poly2)

{

polynomial ptr1,ptr2,new\_poly,res\_poly = Createhead();

float sum\_num;

printf("\nDisplaying First Polynomial:");

display\_poly(poly1);

printf("\nDisplaying Second Polynomial:");

display\_poly(poly2);

ptr1 = poly1->next;

ptr2 = poly2->next;

while(ptr1 != NULL && ptr2 != NULL)

{

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

{ sum\_num = ptr1 -> coeff + ptr2 -> coeff;

res\_poly=Insert\_term(res\_poly, sum\_num, ptr1 -> exp);

ptr1=ptr1->next;

ptr2=ptr2->next;

}

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

{ res\_poly=Insert\_term(res\_poly,ptr1->coeff,ptr1 -> exp);

ptr1=ptr1->next;

}

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

{ res\_poly=Insert\_term(res\_poly,ptr2 ->coeff,ptr2->exp);

ptr2 = ptr2 -> next;

}

}

while(ptr2 != NULL)

{

res\_poly = Insert\_term(res\_poly, ptr2 -> coeff, ptr2 -> exp);

ptr2 = ptr2 -> next;

}

while(ptr1 != NULL)

{

res\_poly = Insert\_term(res\_poly, ptr1 -> coeff, ptr1 -> exp);

ptr1 = ptr1 -> next;

}

return res\_poly;

}

polynomial mul\_poly(polynomial h1,polynomial h2)

{

float res=0;

polynomial temp1,temp2,res\_poly,h3;

printf("\nDisplaying First Polynomial:");

display\_poly(h1);

printf("\nDisplaying Second Polynomial:");

display\_poly(h2);

h3=Createhead();

res\_poly=Createhead();

temp1=h1->next;

while(temp1!=NULL)

{ temp2=h2->next;

while(temp2!=NULL)

{

h3=Insert\_term(h3,temp1->coeff\*temp2->coeff,temp1->exp+temp2->exp);

temp2=temp2->next;

}

temp1=temp1->next;

}

printf("\nDisplaying Initial Result of Product:");

display\_poly(h3);

temp1=h3->next;

while(temp1!=NULL)

{ temp2=temp1->next;

res=0;

while(temp2!=NULL)

{

if(temp1->exp==temp2->exp)

res += temp2->coeff;

temp2=temp2->next;

}

if(search(res\_poly,temp1->exp)==0) //temp1->exp not found

res\_poly=Insert\_term(res\_poly,res+(temp1->coeff),temp1->exp);

temp1=temp1->next;

}

return res\_poly;

}

int search(polynomial h,int val)

{ polynomial tmp=h->next;

while(tmp!=NULL)

{

if(tmp->exp==val)

return 1;

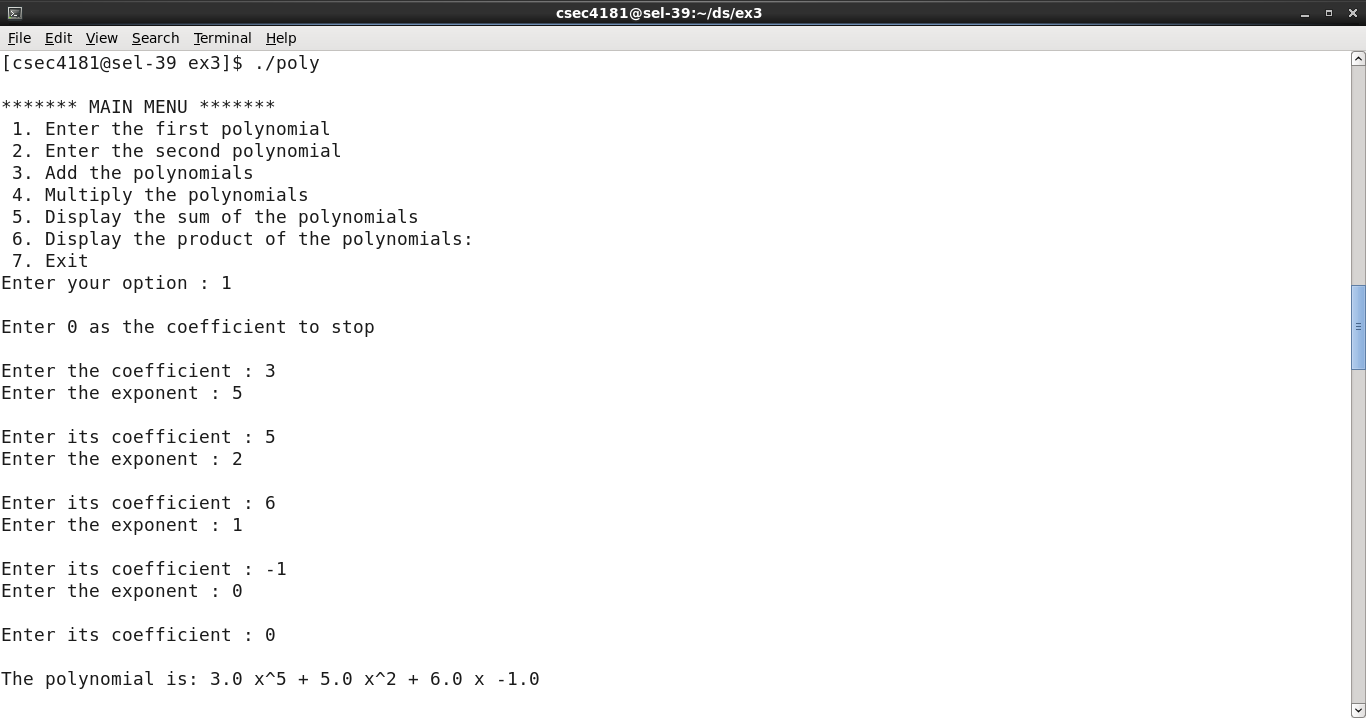
tmp=tmp->next;

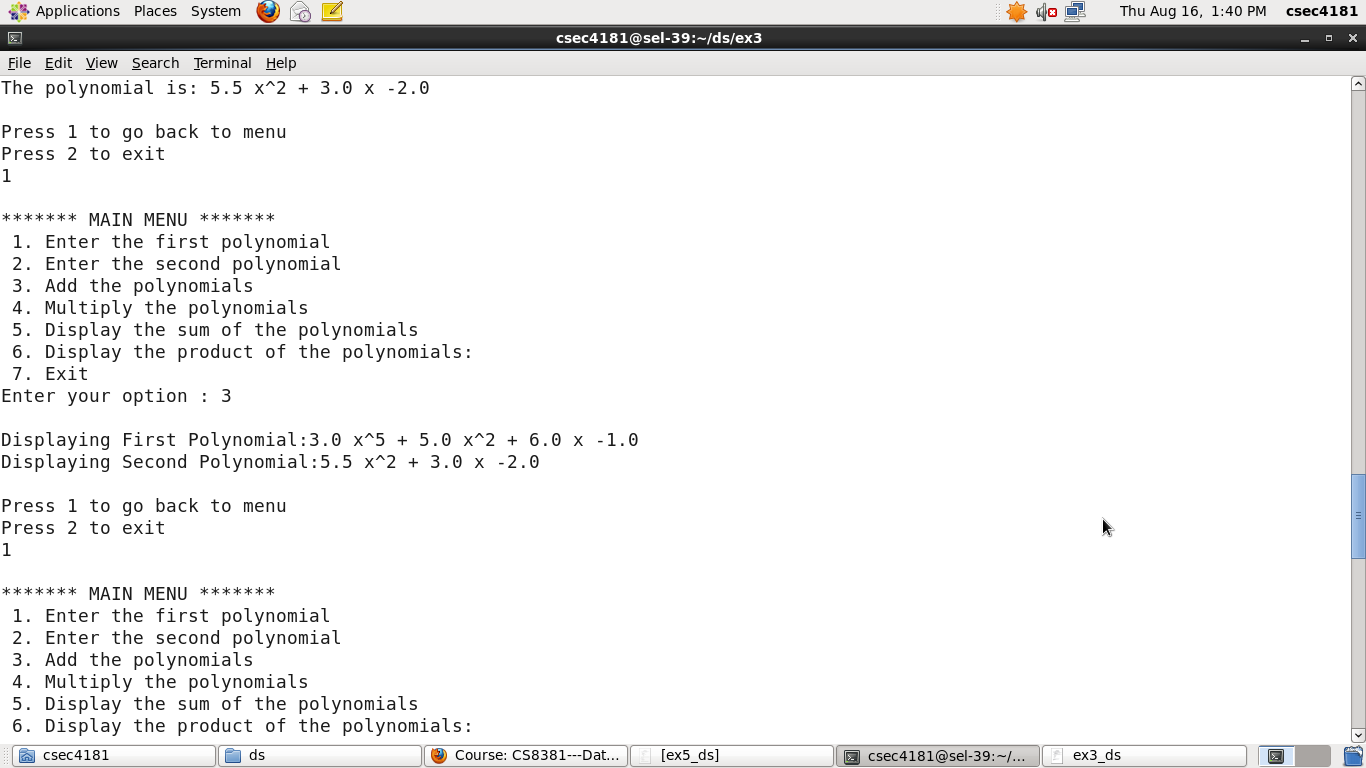
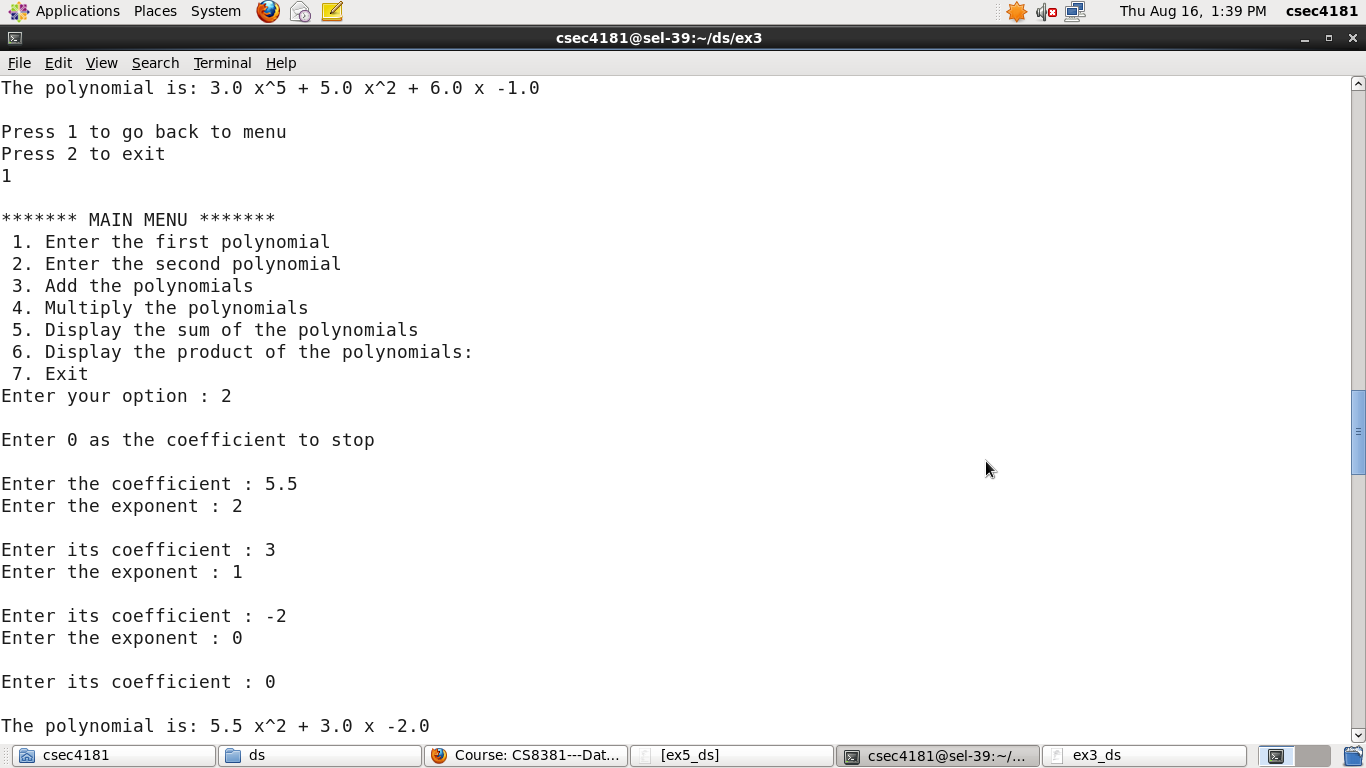
}

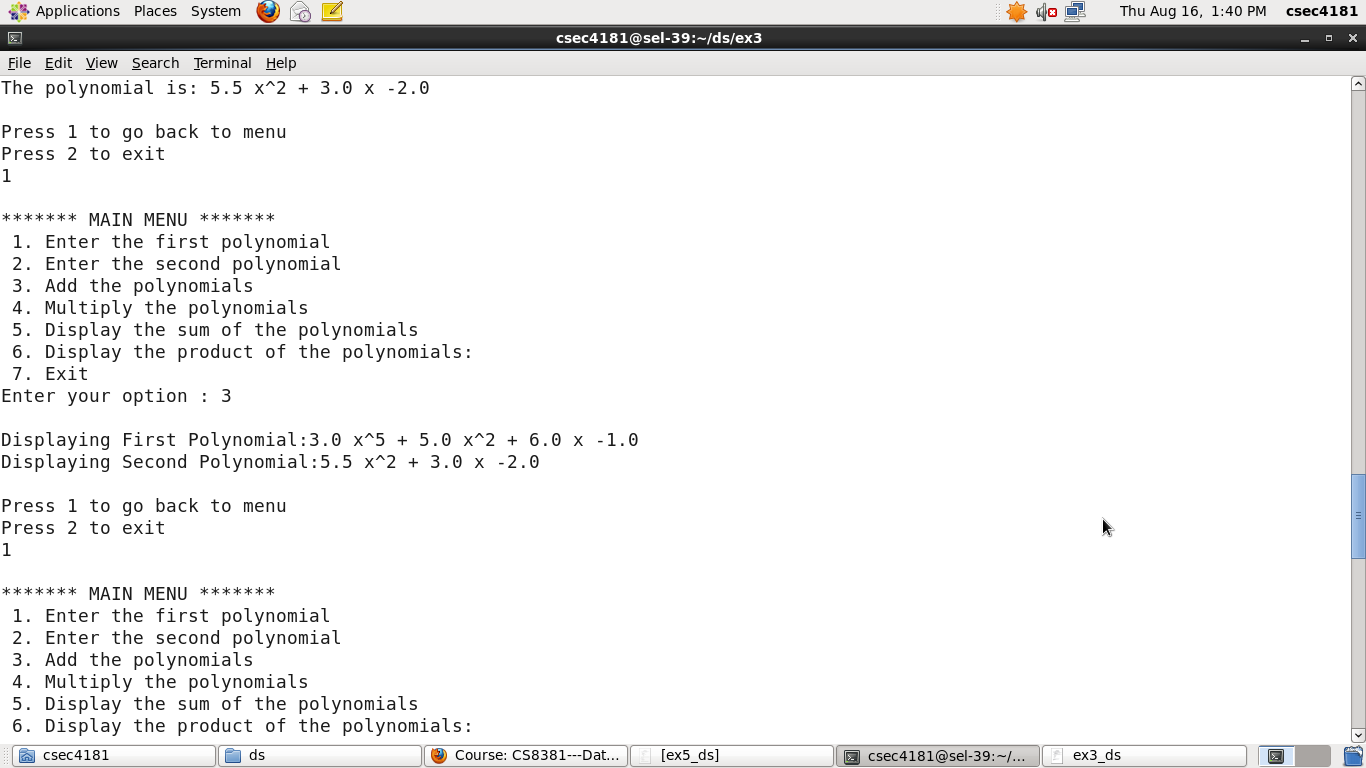
return 0;

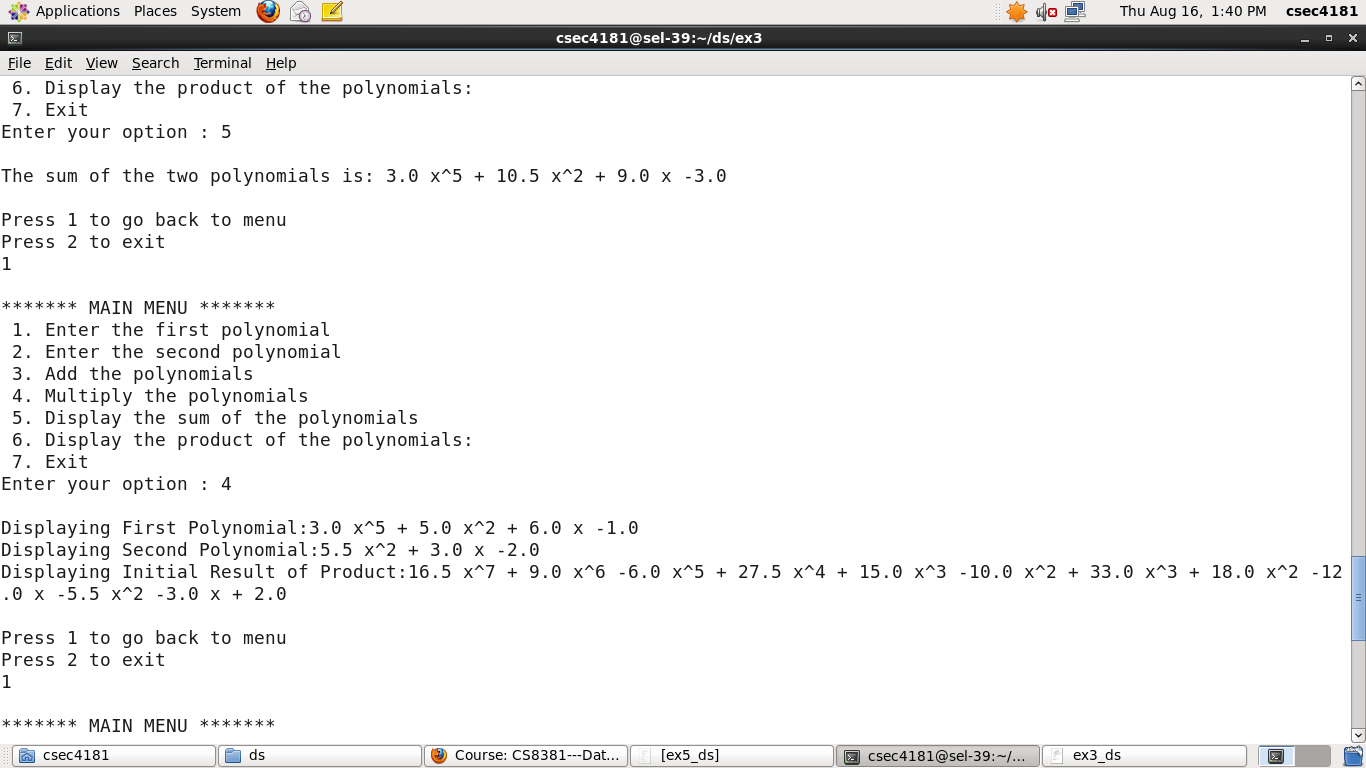
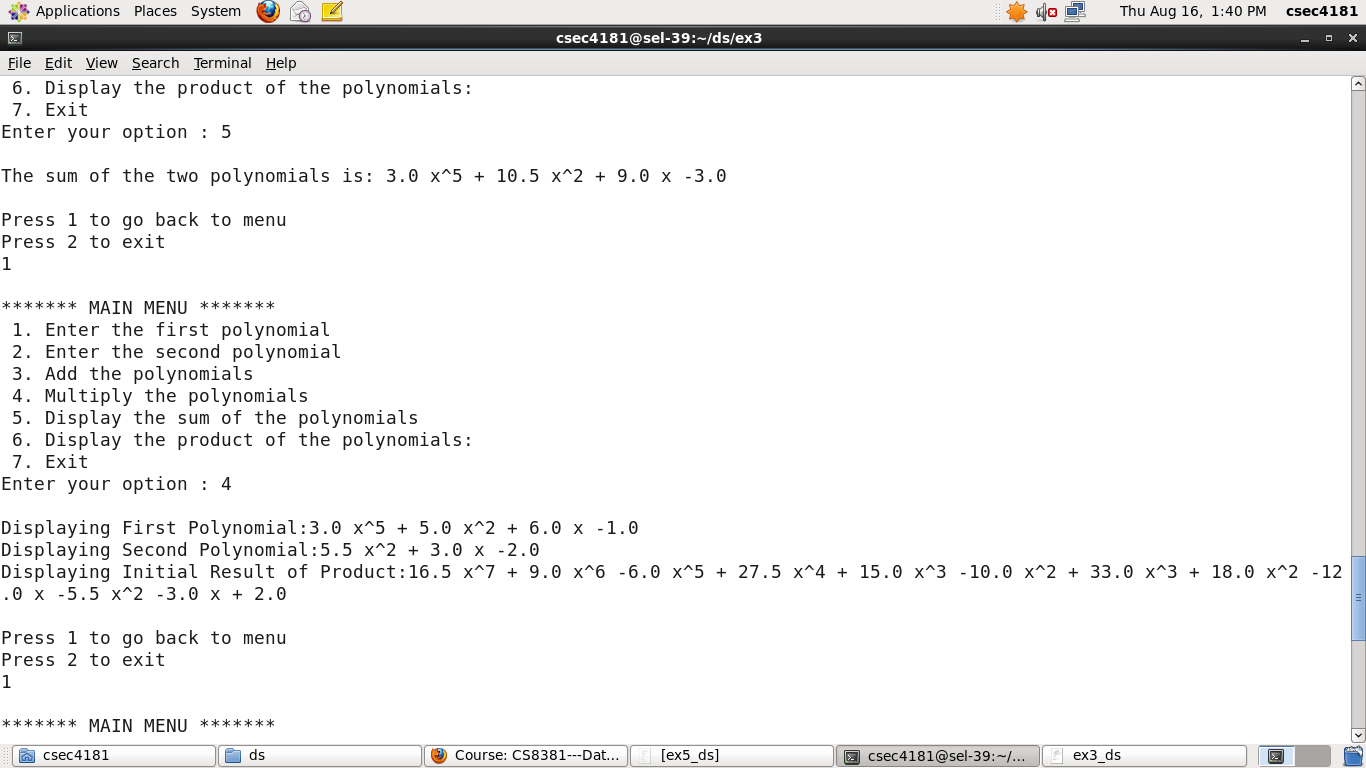
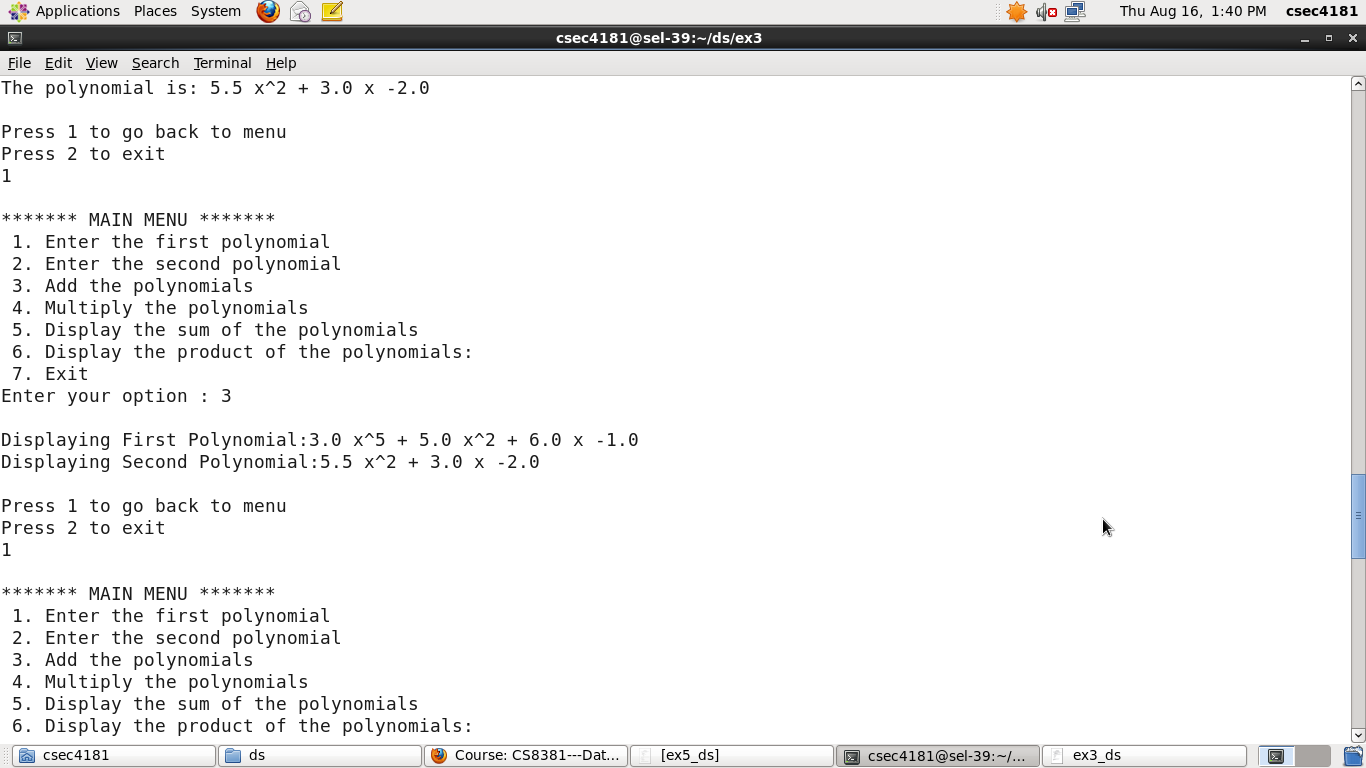
}

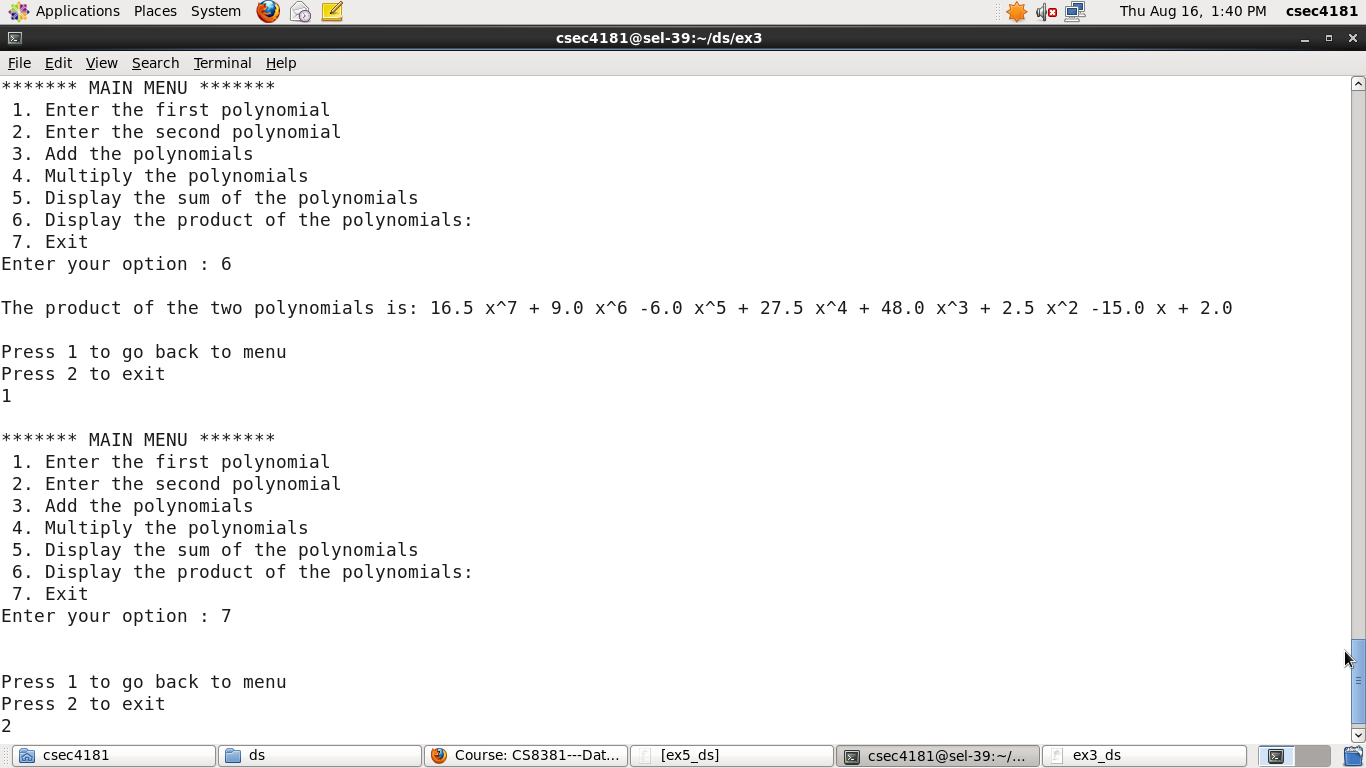
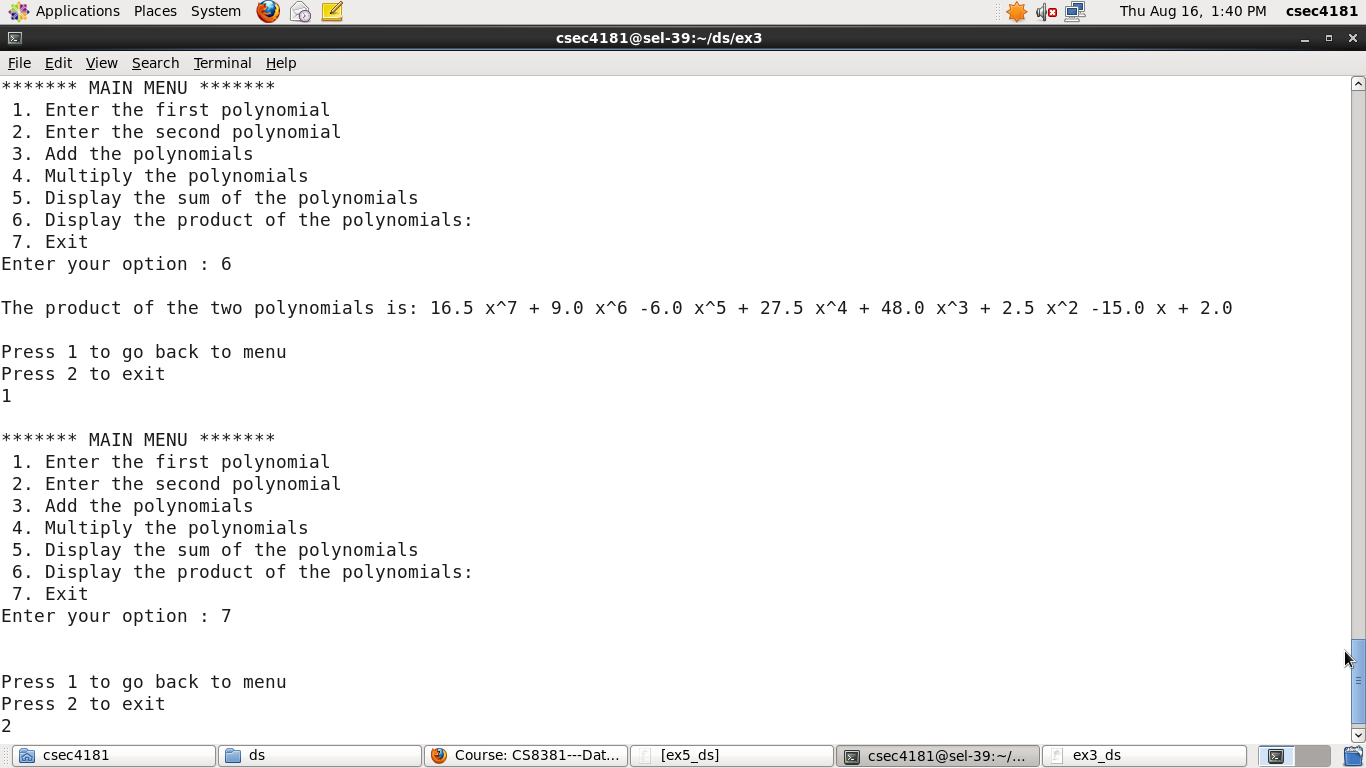
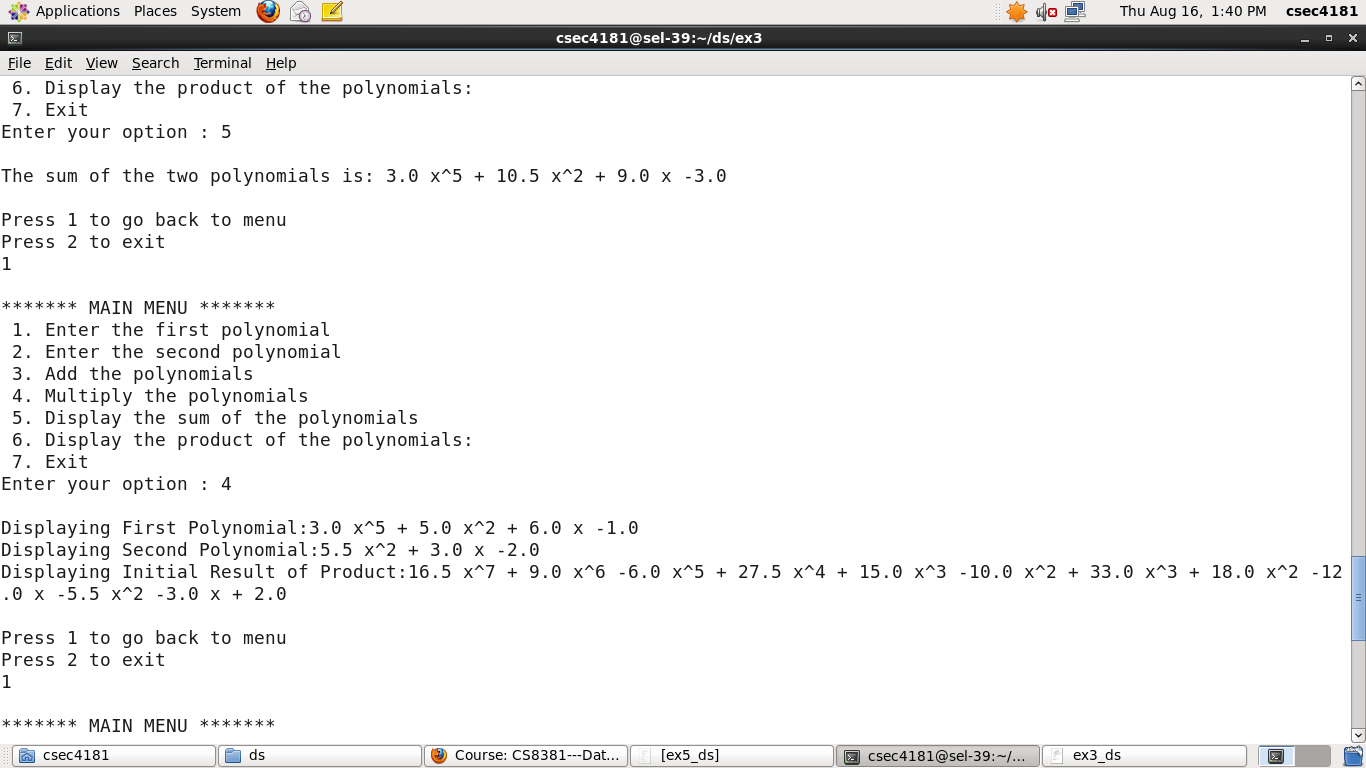
**OUTPUT:**

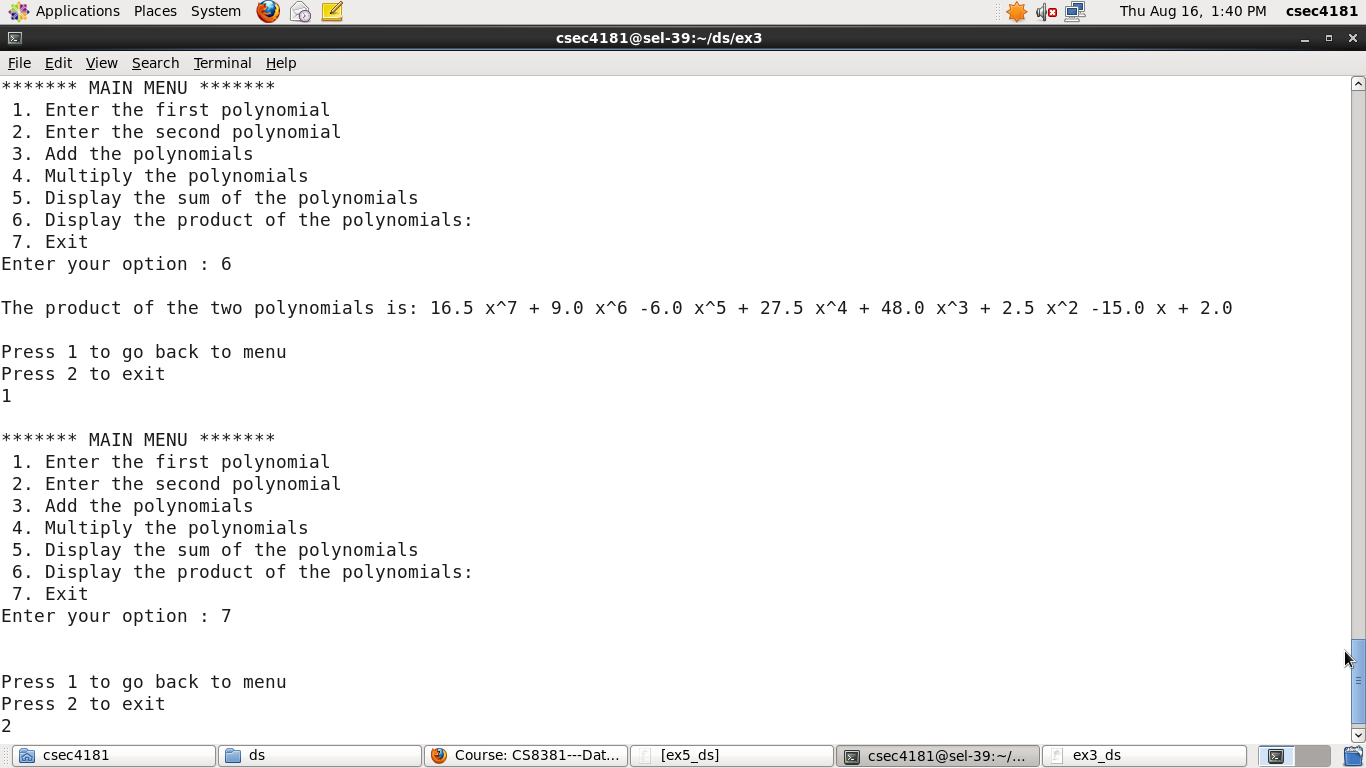


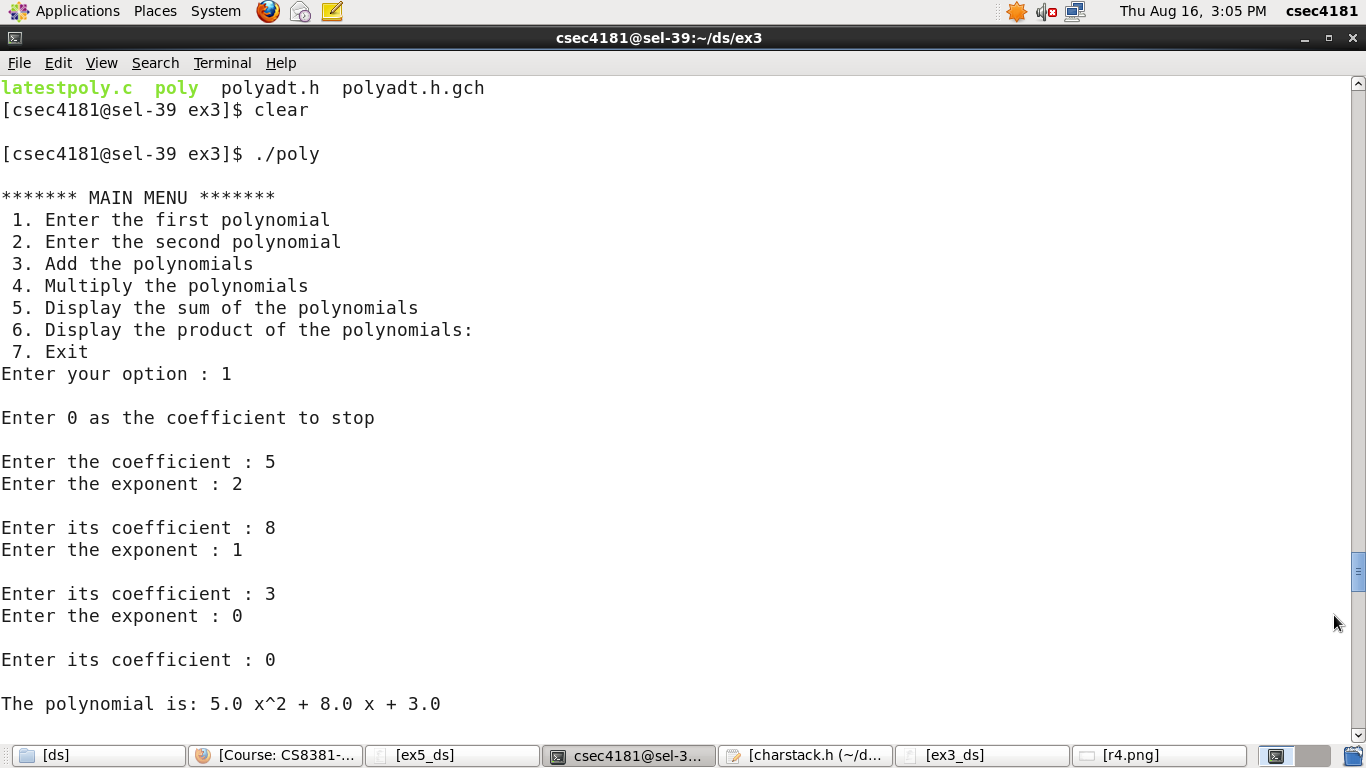


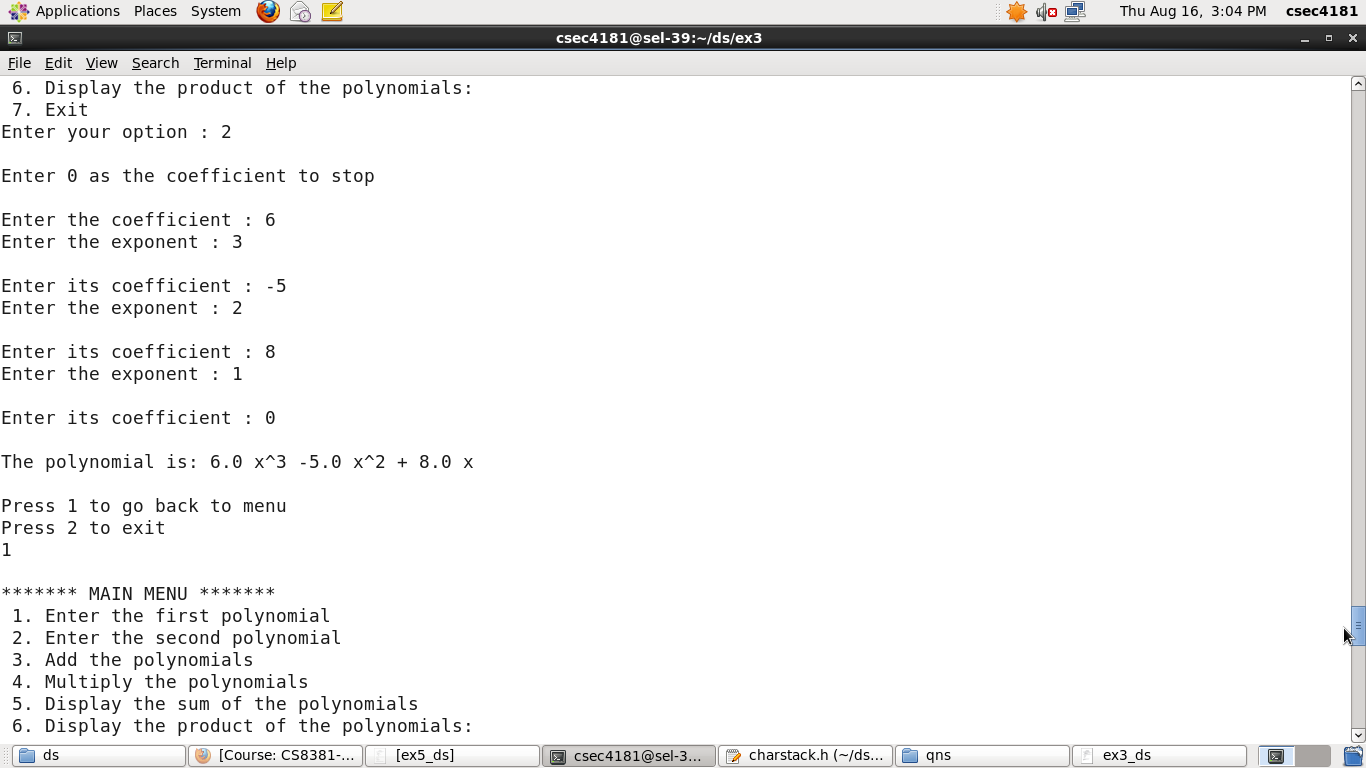


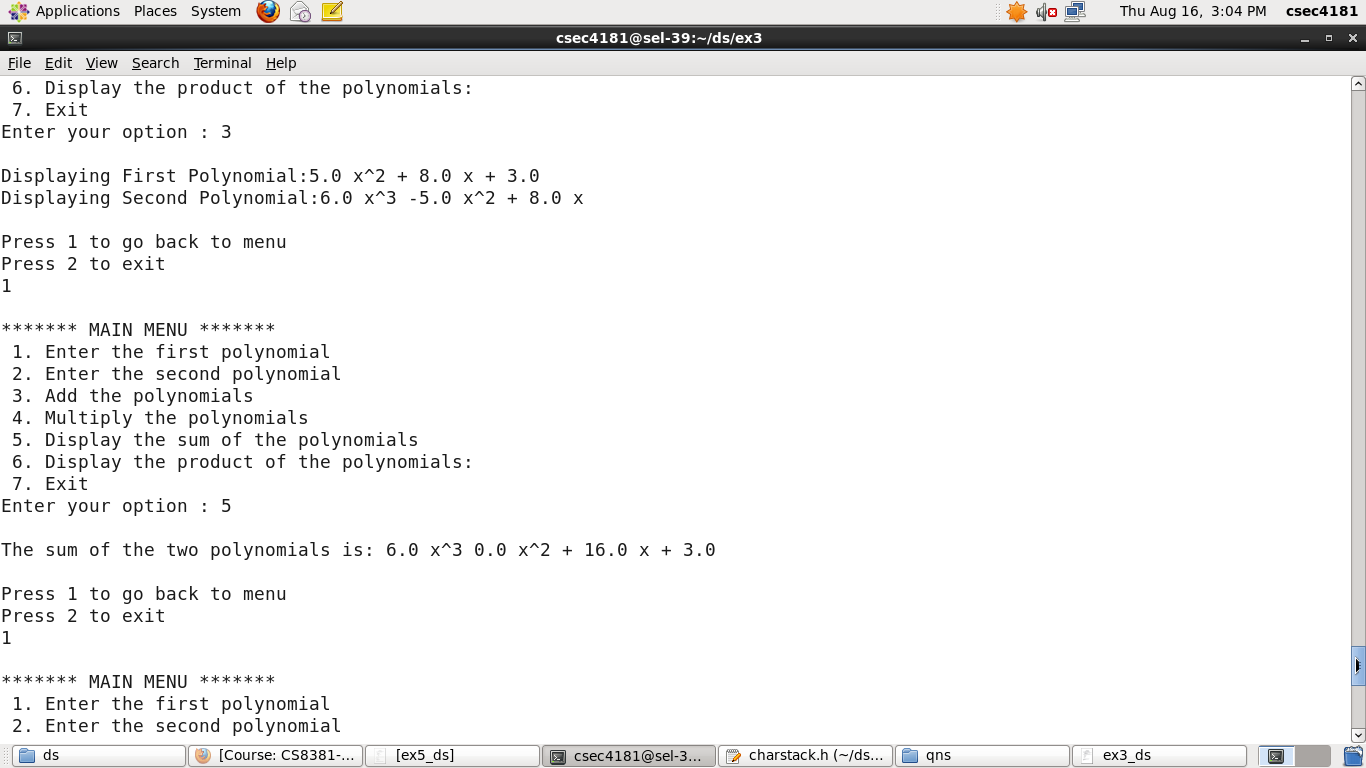


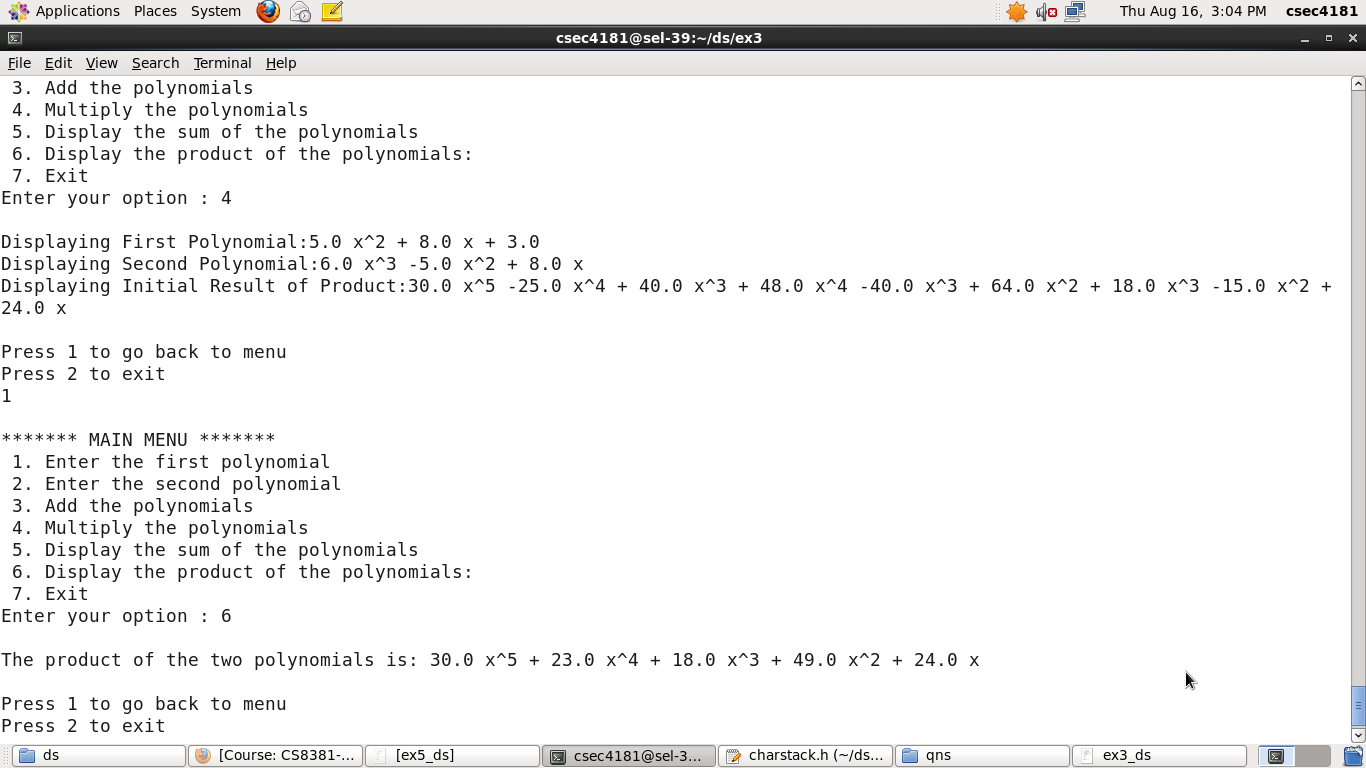


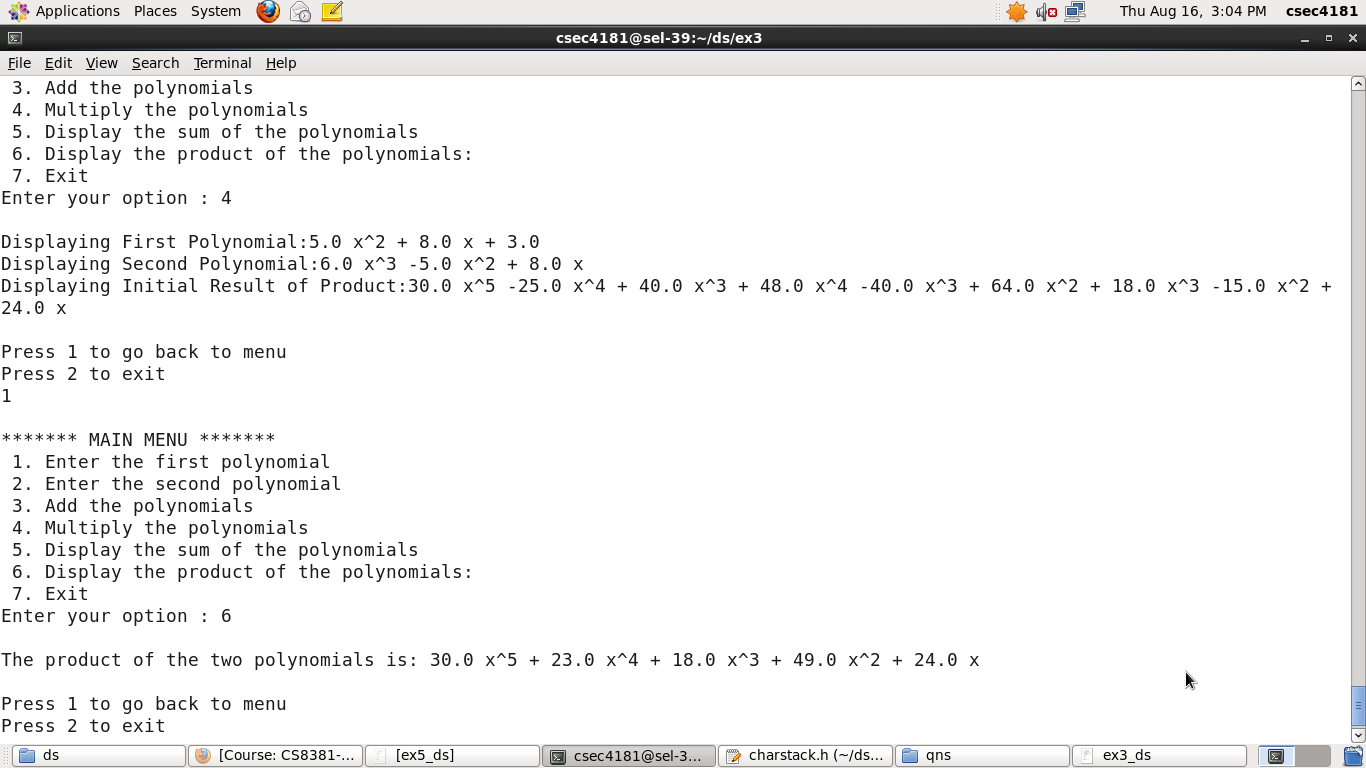


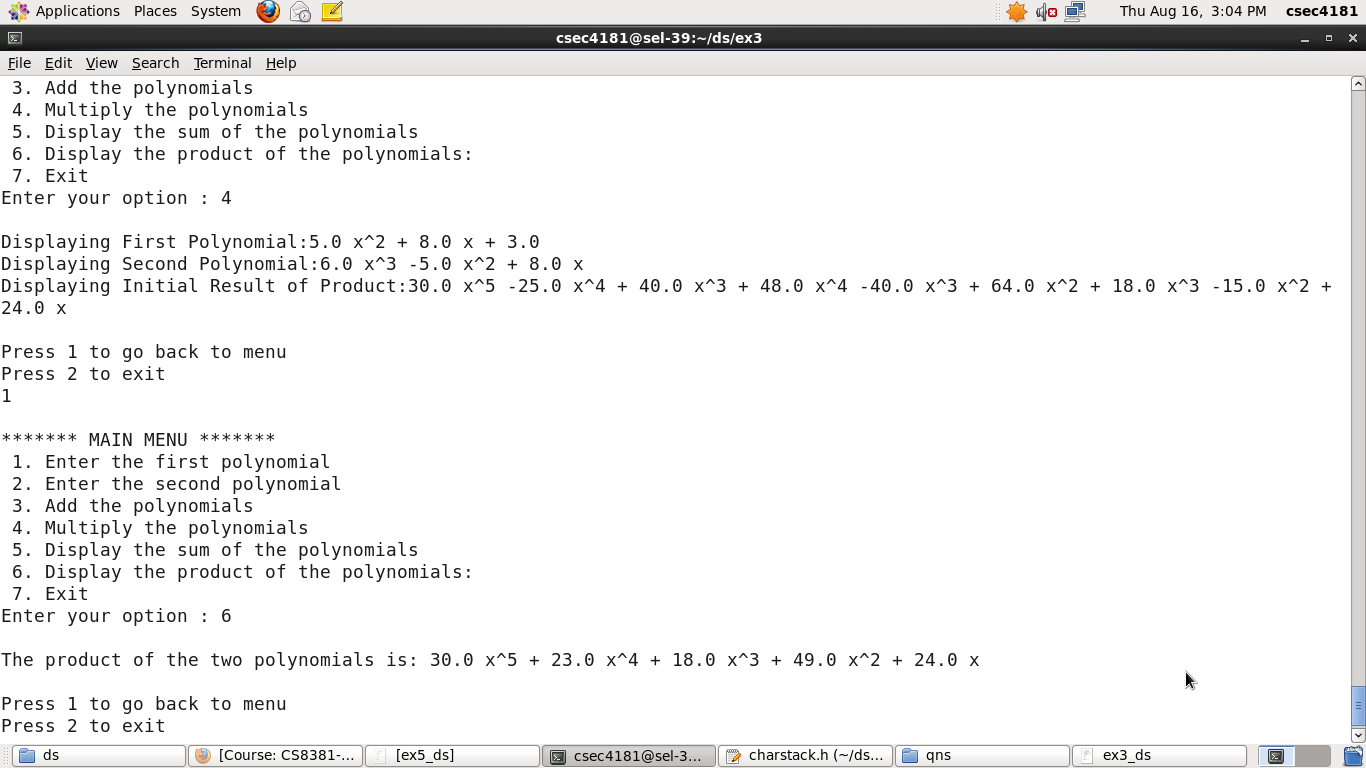


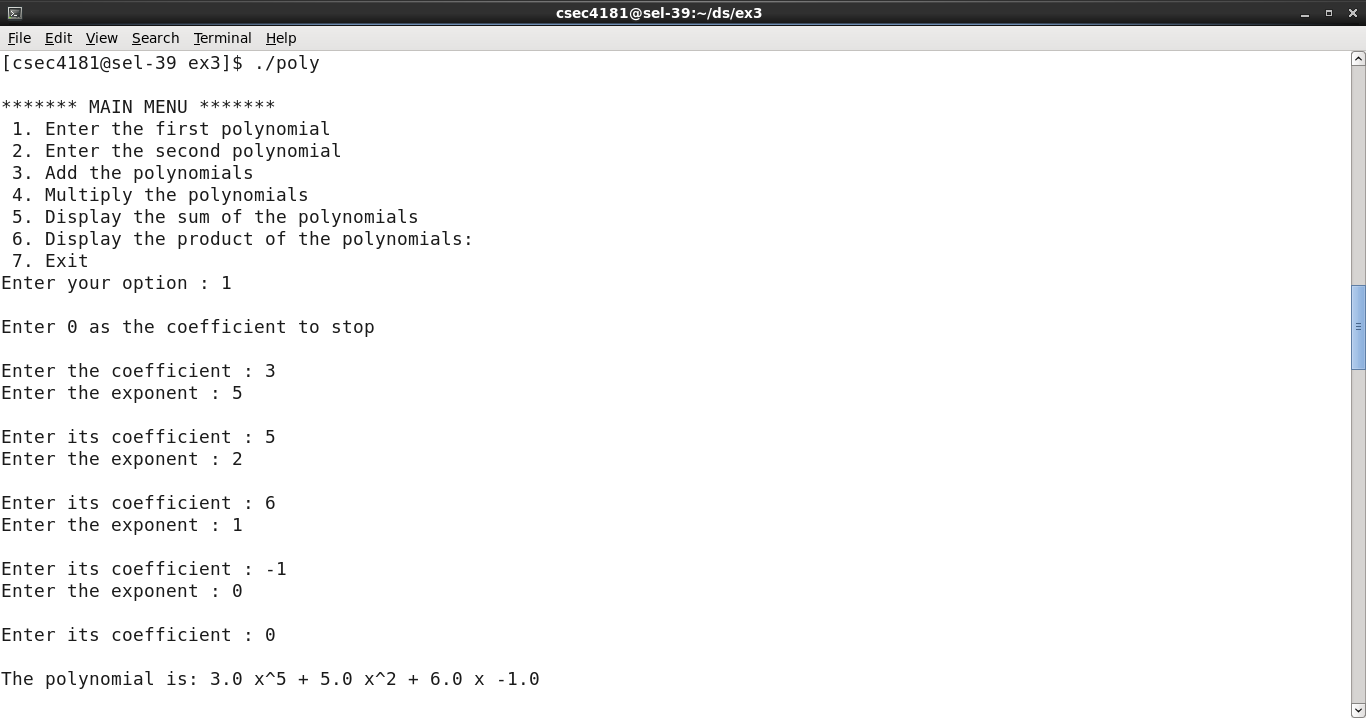


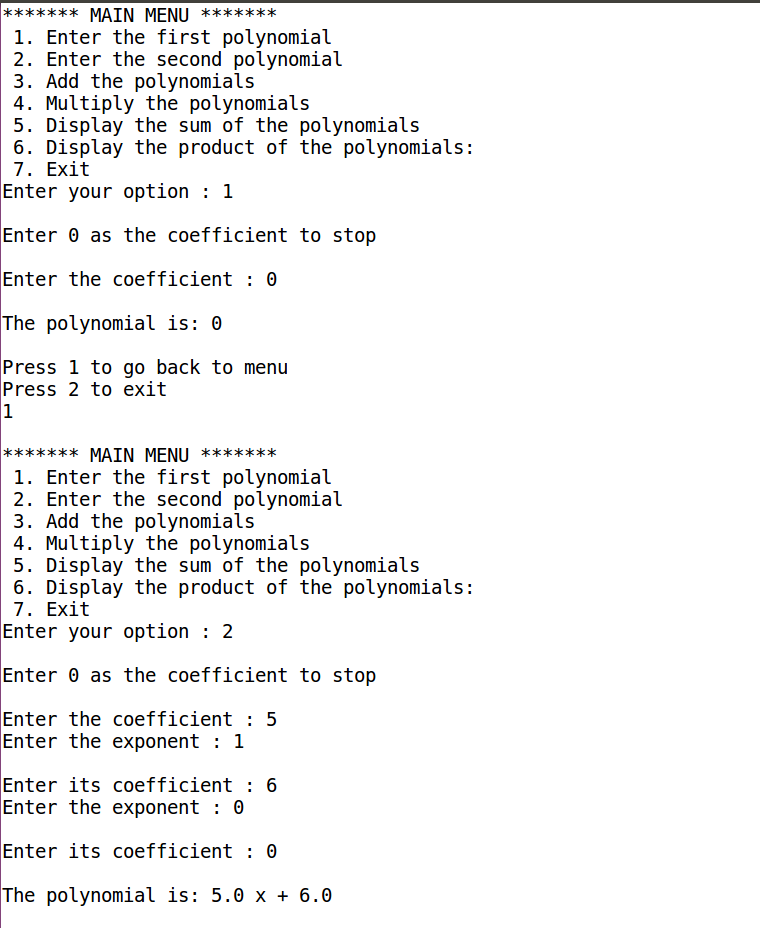


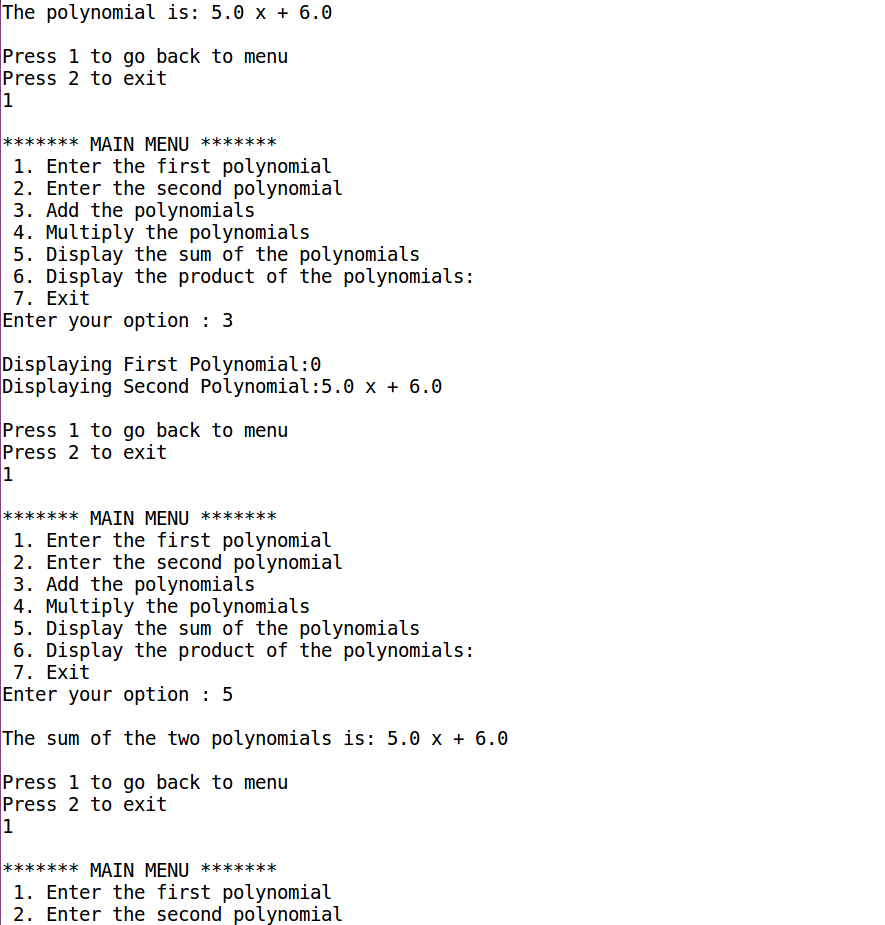


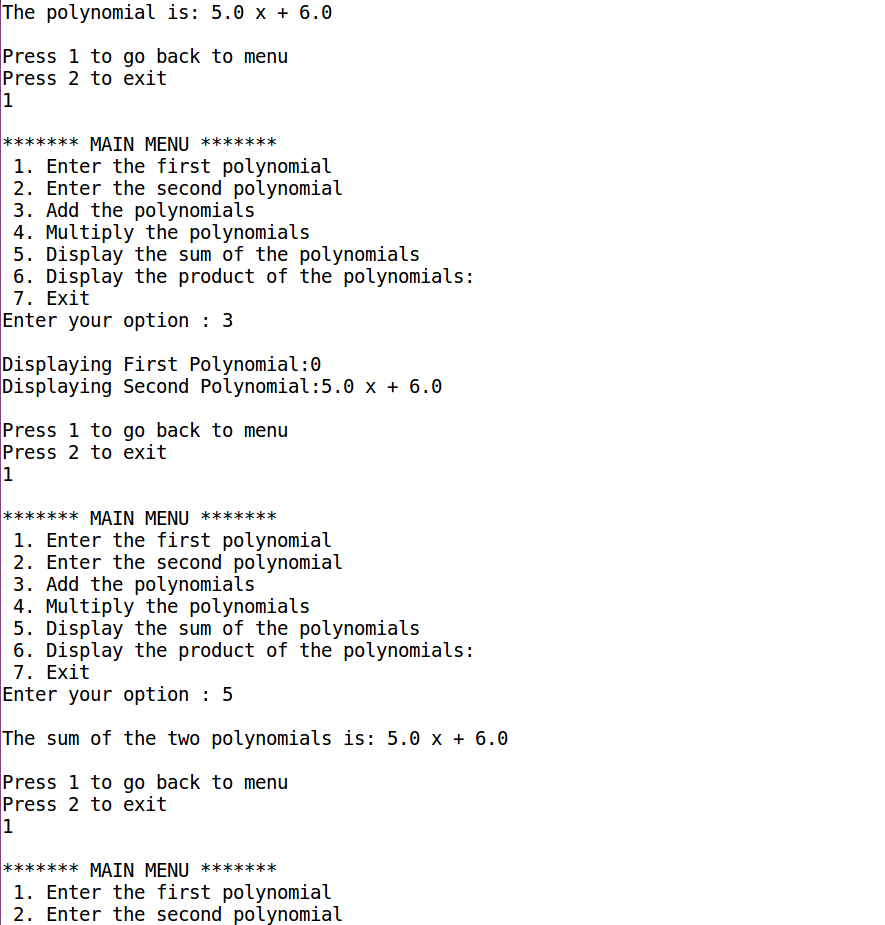


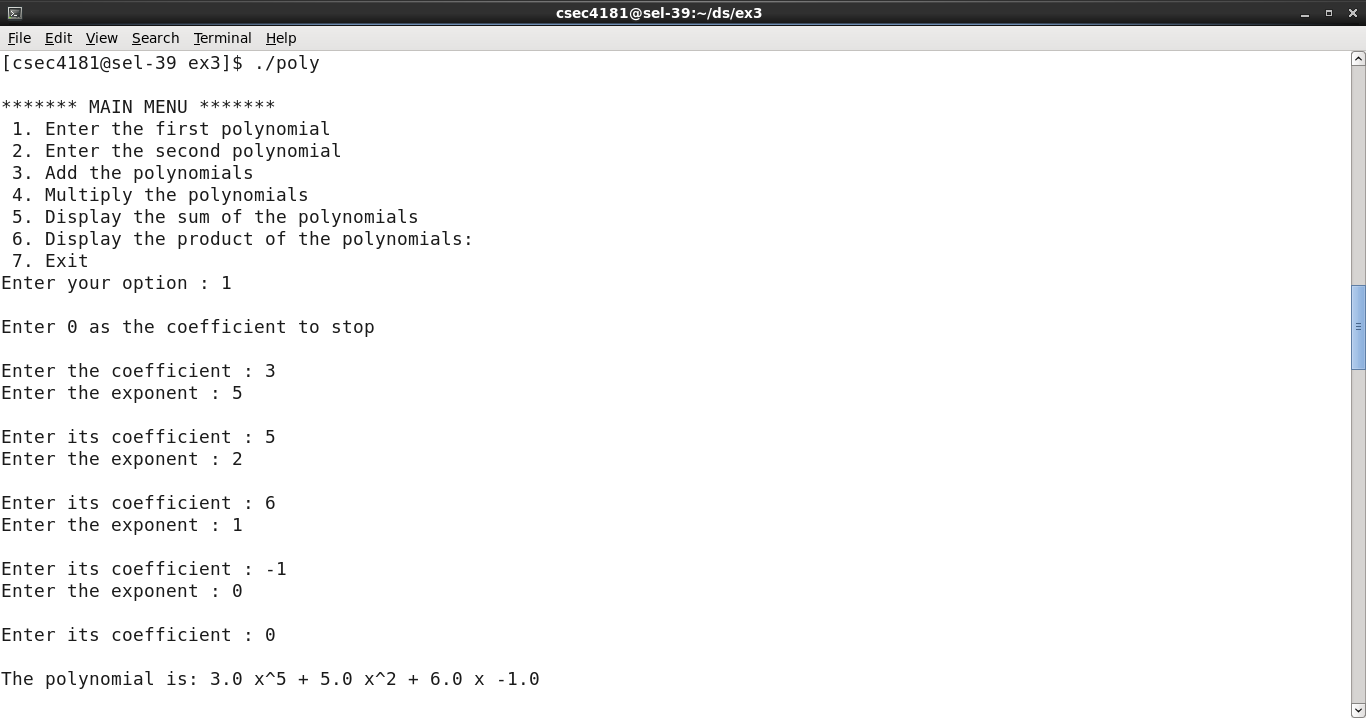


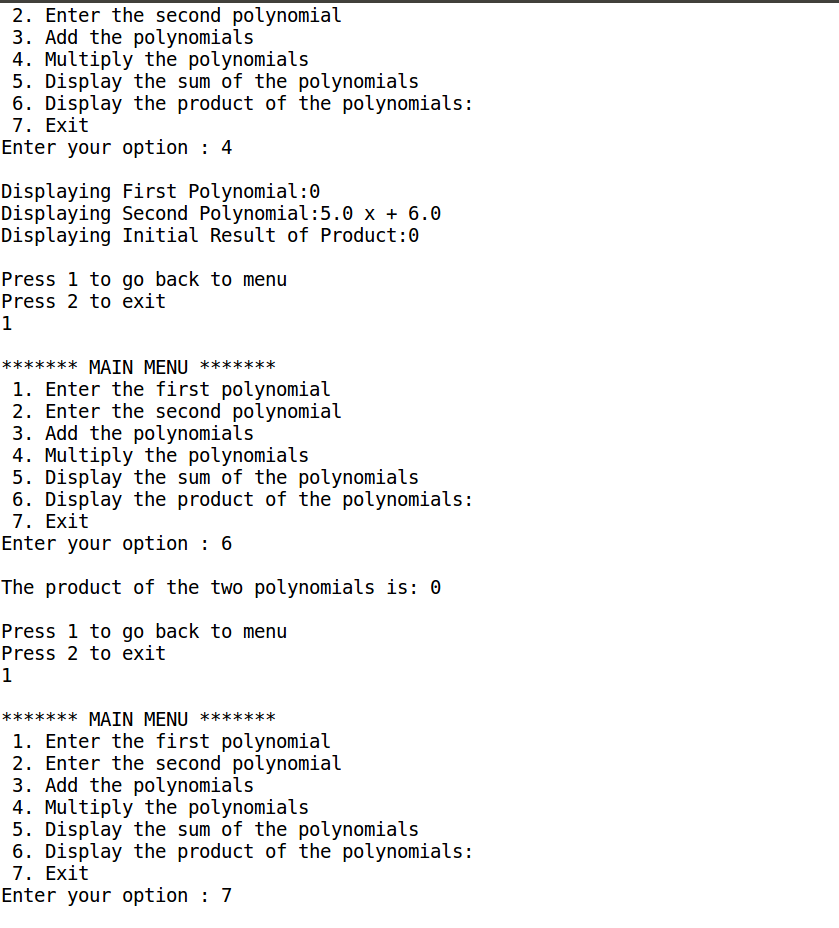
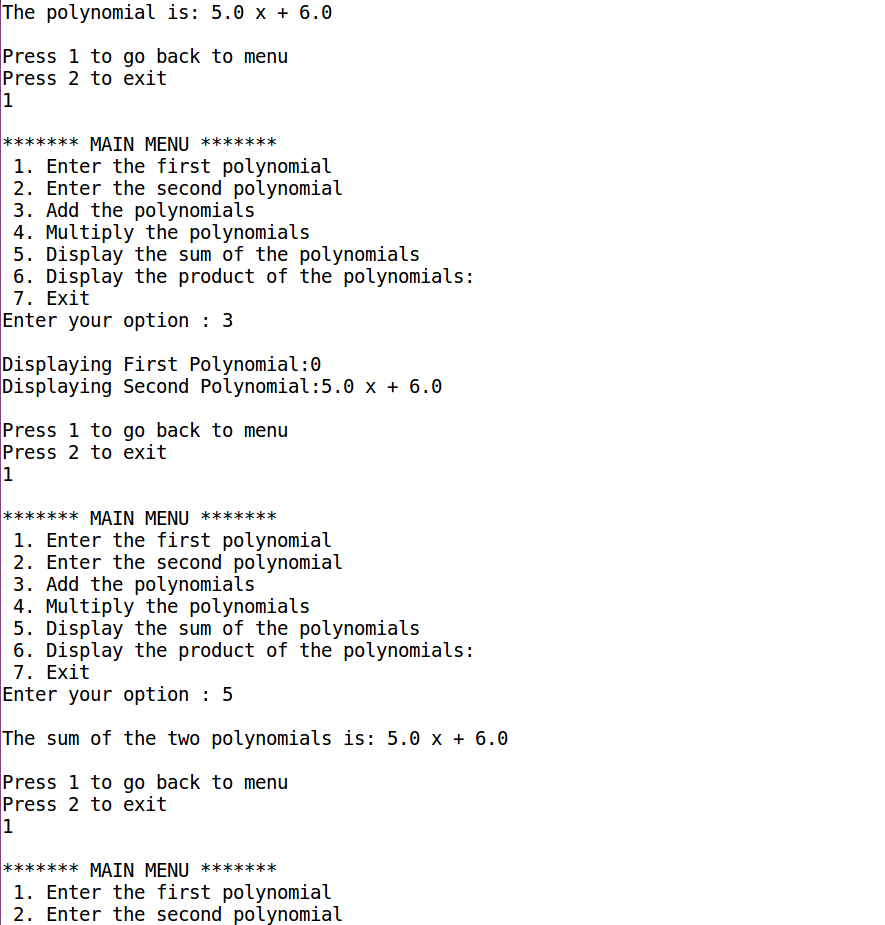


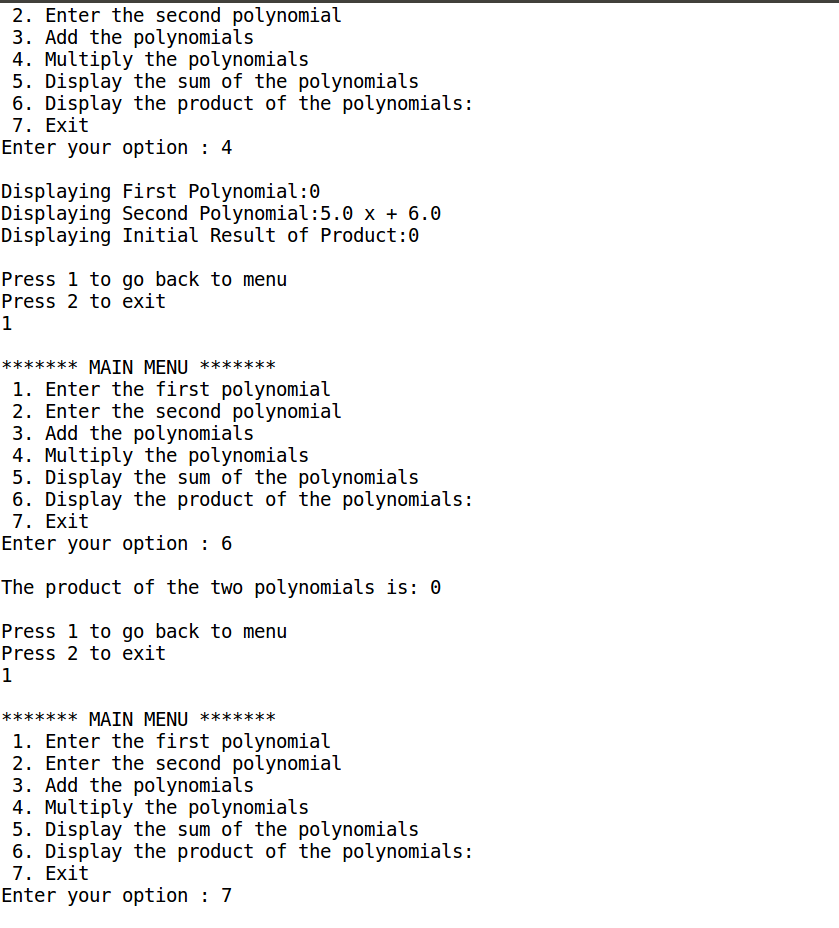


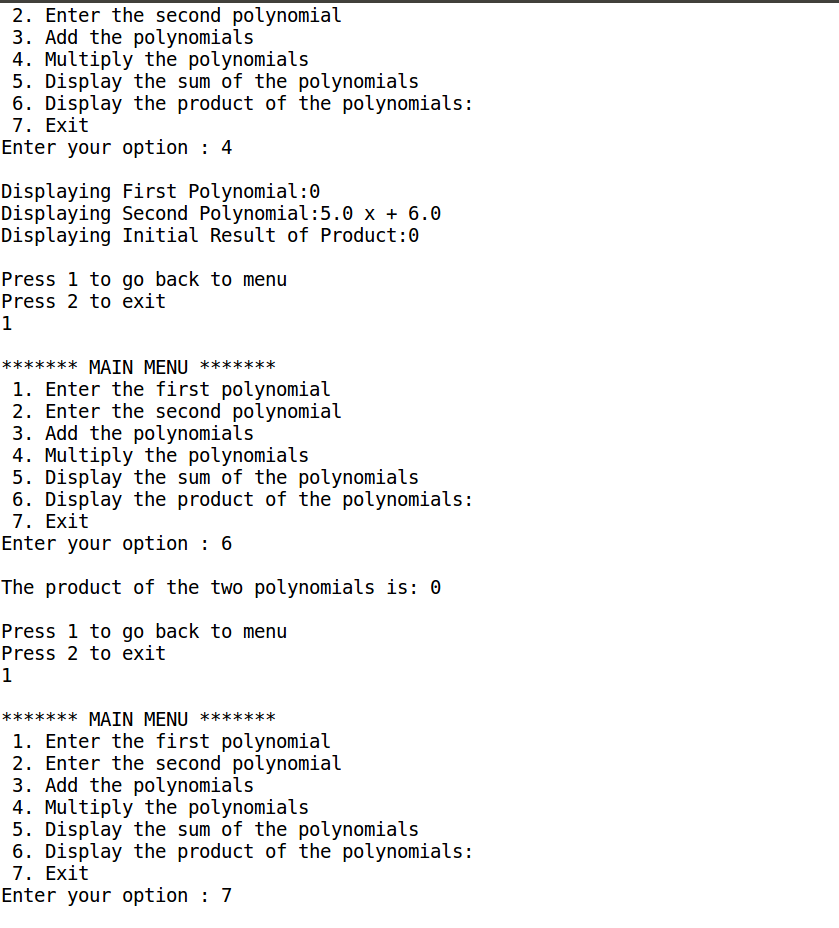












**PROGRAM CODE:**

#include<stdio.h>

#include<stdlib.h>

#include "stackadt.h"

void main()

{

int i,flag=0;

char temp[10];

stack s1=createstack();

printf("Enter the word: ");

scanf("%s", temp);

for(i=0; temp[i]!='\0'; i++)

push(temp[i], s1);

for(i=0; temp[i]!='\0'; i++)

if(temp[i] != return\_top(s1))

{

flag = 1;

break;

}

if(flag == 0)

printf("The given word %s is a palindrome!\n" ,temp);

else

printf("The given word %s is not a palindrome!\n",temp);

}

**stackadt header file**

typedef struct node \*ptrtonode;

typedef ptrtonode stack;

struct node

{

char element;

ptrtonode next;

};

int isempty(stack s)

{

return s->next==NULL;

}

stack createstack( )

{

stack s=(stack)malloc(sizeof(struct node));

if(s==NULL)

printf("Out of space");

return s;

}

void pop(stack s)

{

ptrtonode tmpcell;

if(isempty(s))

printf("\nStack is empty");

else

{

tmpcell = s->next;

s->next = s->next->next;

free(tmpcell);

}

}

void dispose(stack s)

{

if(s==NULL)

printf("Must use createstack first");

else

{ while(!isempty(s))

pop(s);

free(s);

printf("Stack disposed" );

}

}

void push(char x, stack s)

{

ptrtonode tmpcell=(stack)malloc(sizeof(struct node));

if(tmpcell==NULL)

printf("Out of space");

else

{

tmpcell->element = x;

tmpcell->next = s->next;

s->next = tmpcell;

}

}

int return\_top(stack s)

{

char x;

if(!isempty(s))

{

x=s->next->element;

pop(s);

return x;

}

printf("\nStack is empty");

return 0;

}

void display(stack s)

{

ptrtonode temp=s->next;

printf("\nThe elements are: ");

while (temp!=NULL)

{ printf("%c",temp->element);

temp=temp->next;

}

}

**OUPTUT:**

