**CODE:-**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct node node;

struct node

{

int coef;

int deg;

node \*link;

};

void remdupl(node \*start)

{

node \*p, \*q, \*temp;

p = start;

while (p != NULL && p->link != NULL)

{

q = p;

while (q->link != NULL)

{

if (p->deg == q->link->deg)

{

p->coef = p->coef + q->link->coef;

temp = q->link;

q->link = q->link->link;

free(temp);

}

else

q = q->link;

}

p = p->link;

}

}

node \*create(node \*start, int polyno)

{

node \*temp, \*p;

int n;

printf("Enter the terms of the polynomial in descending powers.\n");

printf("Enter the number of terms of the polynomial %d: ", polyno);

scanf("%d", &n);

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

{

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

printf("Enter the coefficient: ");

scanf("%d", &temp->coef);

printf("Enter the power: ");

scanf("%d", &temp->deg);

temp->link = NULL;

if (start == NULL)

start = temp;

else

{

p = start;

while (p->link != NULL)

p = p->link;

p->link = temp;

}

}

remdupl(start);

return start;

}

node \*create\_from\_file(node \*start, FILE \*fp)

{

node \*temp, \*p;

int n;

fscanf(fp, "%d ", &n);

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

{

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

fscanf(fp, "%d %d ", &temp->coef, &temp->deg);

temp->link = NULL;

if (start == NULL)

start = temp;

else

{

p = start;

while (p->link != NULL)

p = p->link;

p->link = temp;

}

}

return start;

}

void display(node \*start)

{

node \*p;

p = start;

while (p != NULL)

{

printf("%dx^%d", p->coef, p->deg);

p = p->link;

if (p != NULL)

printf(" + ");

}

printf("\n");

}

node \*add(node \*start1, node \*start2)

{

node \*start3, \*p, \*q, \*temp;

start3 = NULL;

p = start1;

q = start2;

while (p != NULL && q != NULL)

{

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

temp->link = NULL;

if (p->deg > q->deg)

{

temp->deg = p->deg;

temp->coef = p->coef;

p = p->link;

}

else if (p->deg < q->deg)

{

temp->deg = q->deg;

temp->coef = q->coef;

q = q->link;

}

else

{

temp->coef = p->coef + q->coef;

temp->deg = p->deg;

p = p->link;

q = q->link;

}

if (start3 == NULL)

start3 = temp;

else

{

node \*r = start3;

while (r->link != NULL)

r = r->link;

r->link = temp;

}

}

while (p != NULL)

{

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

temp->coef = p->coef;

temp->deg = p->deg;

temp->link = NULL;

p = p->link;

if (start3 == NULL)

start3 = temp;

else

{

node \*r = start3;

while (r->link != NULL)

r = r->link;

r->link = temp;

}

}

while (q != NULL)

{

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

temp->coef = q->coef;

temp->deg = q->deg;

temp->link = NULL;

q = q->link;

if (start3 == NULL)

start3 = temp;

else

{

node \*r = start3;

while (r->link != NULL)

r = r->link;

r->link = temp;

}

}

remdupl(start3);

return start3;

}

node \*multiply(node \*start1, node \*start2)

{

node \*start3, \*p, \*q, \*temp;

start3 = NULL;

p = start1;

while (p != NULL)

{

q = start2;

while (q != NULL)

{

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

temp->coef = p->coef \* q->coef;

temp->deg = p->deg + q->deg;

temp->link = NULL;

if (start3 == NULL)

start3 = temp;

else

{

node \*r = start3;

while (r->link != NULL)

r = r->link;

r->link = temp;

}

q = q->link;

}

p = p->link;

}

remdupl(start3);

return start3;

}

node \*addatbeg(node \*start, node \*p)

{

p->link = start;

start = p;

return start;

}

node \*addatend(node \*start, node \*p)

{

node \*temp = start;

if (temp == NULL)

{

start = p;

return start;

}

while (temp->link != NULL)

temp = temp->link;

temp->link = p;

return start;

}

struct node \*powerdel(struct node \*start, int exp)

{

struct node \*p = start;

if (start == NULL)

return start;

else if (start->deg == exp)

{

p = start;

start = start->link;

free(p);

return start;

}

while (p->link != NULL)

{

if (p->link->deg == exp)

{

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

temp = p->link;

p->link = temp->link;

free(temp);

return start;

}

p = p->link;

}

}

node \*adder(node \*start, node \*p)

{

if ((start == NULL) || (p->deg > start->deg))

{

start = addatbeg(start, p);

return start;

}

node \*temp = start;

while ((temp->link != NULL) && (p->deg < temp->link->deg))

temp = temp->link;

if (temp->link == NULL)

{

if (p->deg > temp->deg)

{

p->link = temp;

start = p;

}

else if (p->deg == temp->deg)

{

temp->coef += p->coef;

}

else if (p->deg < temp->deg)

{

temp->link = p;

}

return start;

}

else

{

if (p->deg == temp->link->deg)

temp->link->coef += p->coef;

else if (p->deg > temp->link->deg)

{

p->link = temp->link;

temp->link = p;

}

}

return start;

}

node \*modify(node \*start)

{

printf("Press 1 to insert a term, 2 to delete a term.\n");

int ch, coef, deg;

scanf("%d", &ch);

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

switch (ch)

{

case 1:

printf("Enter the coefficient : ");

scanf("%d", &(temp->coef));

printf("Enter the power : ");

scanf("%d", &(temp->deg));

temp->link = NULL;

adder(start, temp);

break;

case 2:

printf("Enter the power of the term to be deleted.\n");

scanf("%d", &deg);

start = powerdel(start, deg);

break;

default:

printf("Invalid input.\n");

}

return start;

}

int main()

{

node \*start1, \*start2;

while (1)

{

printf("Enter 1 to enter polynomials.\n");

printf("Enter 2 to display polynomials.\n");

printf("Enter 3 to add polynomials.\n");

printf("Enter 4 to multiply polynomials.\n");

printf("Enter 5 to modify a polynomial.\n");

printf("Enter 6 to get polynomials from file.\n");

printf("Enter 7 to exit.\n");

int choice;

scanf("%d", &choice);

switch (choice)

{

case 1:

node \*start1 = NULL, \*start2 = NULL;

start1 = create(start1, 1);

start2 = create(start2, 2);

break;

case 2:

printf("Polynomial 1: ");

display(start1);

printf("Polynomial 2: ");

display(start2);

break;

case 3:

printf("Addition Result is: ");

display(add(start1, start2));

break;

case 4:

printf("Multiplication Result is: ");

display(multiply(start1, start2));

break;

case 5:

printf("Which polynomial should be modified: 1 or 2?\n");

int polynum;

scanf("%d", &polynum);

if (polynum == 1)

modify(start1);

else

modify(start2);

break;

case 6:

start1 = NULL, start2 = NULL;

FILE \*fp;

fp = fopen("poly.txt", "r");

start1 = create\_from\_file(start1, fp);

start2 = create\_from\_file(start2, fp);

fclose(fp);

break;

case 7:

exit(1);

break;

default:

printf("Invalid choice.\n");

break;

}

}

return 0;

}

**OUTPUT:-**

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

1

Enter the terms of the polynomial in descending powers.

Enter the number of terms of the polynomial 1: 3

Enter the coefficient: 5

Enter the power: 4

Enter the coefficient: 2

Enter the power: 2

Enter the coefficient: 7

Enter the power: 0

Enter the terms of the polynomial in descending powers.

Enter the number of terms of the polynomial 2: 2

Enter the coefficient: 7

Enter the power: 1

Enter the coefficient: 8

Enter the power: 0

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

2

Polynomial 1: 5x^4 + 2x^2 + 7x^0

Polynomial 2: 7x^1 + 8x^0

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

3

Addition Result is: 5x^4 + 2x^2 + 7x^1 + 15x^0

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

4

Multiplication Result is: 35x^5 + 40x^4 + 14x^3 + 16x^2 + 49x^1 + 56x^0

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

5

Which polynomial should be modified: 1 or 2?

1

Press 1 to insert a term, 2 to delete a term.

1

Enter the coefficient : 3

Enter the power : 3

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

2

Polynomial 1: 5x^4 + 3x^3 + 2x^2 + 7x^0

Polynomial 2: 7x^1 + 8x^0

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

5

Which polynomial should be modified: 1 or 2?

1

Press 1 to insert a term, 2 to delete a term.

2

Enter the power of the term to be deleted.

3

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

2

Polynomial 1: 5x^4 + 2x^2 + 7x^0

Polynomial 2: 7x^1 + 8x^0

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

6

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

2

Polynomial 1: 4x^4 + 32x^2 + 4x^1 + 5x^0

Polynomial 2: 3x^1 + 7x^0

Enter 1 to enter polynomials.

Enter 2 to display polynomials.

Enter 3 to add polynomials.

Enter 4 to multiply polynomials.

Enter 5 to modify a polynomial.

Enter 6 to get polynomials from file.

Enter 7 to exit.

7

**CODE:-**

#include <stdio.h>

#include <stdlib.h>

typedef struct node node;

struct node

{

int info;

node \*next;

};

node \*create(node \*start, int setno)

{

node \*temp, \*p;

int num = 0;

printf("Set %d:\n", setno);

printf("Enter the number of elements.\n");

scanf("%d", &num);

printf("Enter the elements.\n");

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

{

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

scanf("%d", &temp->info);

temp->next = NULL;

if (start == NULL)

start = temp;

else

{

p = start;

while (p->next != NULL)

p = p->next;

p->next = temp;

}

}

return start;

}

int search(struct node \*s, int num)

{

while (s != NULL)

{

if (s->info == num)

{

return 1;

}

s = s->next;

}

return 0;

}

void display(node \*start, int setno)

{

node \*p;

p = start;

printf("The elements of set %d are : ", setno);

while (p != NULL)

{

printf("%d ", p->info);

p = p->next;

}

printf("\n");

}

node \*setunion(node \*s1, node \*s2)

{

node \*s3 = NULL;

node \*p = s1;

node \*temp;

while (s1 != NULL)

{

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

temp->info = s1->info;

temp->next = NULL;

if (s3 == NULL)

s3 = temp;

else

{

node \*q = s3;

while (q->next != NULL)

q = q->next;

q->next = temp;

}

s1 = s1->next;

}

while (s2 != NULL)

{

if (!search(p, s2->info))

{

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

temp->info = s2->info;

temp->next = NULL;

node \*r = s3;

while (r->next != NULL)

r = r->next;

r->next = temp;

}

s2 = s2->next;

}

return s3;

}

node \*setintersect(node \*s1, node \*s2)

{

node \*s3 = NULL, \*p, \*q, \*temp;

p = s1;

while (p != NULL)

{

q = s2;

while (q != NULL)

{

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

if (p->info == q->info)

{

temp->info = q->info;

temp->next = NULL;

if (s3 == NULL)

s3 = temp;

else

{

node \*r = s3;

while (r->next != NULL)

r = r->next;

r->next = temp;

}

}

q = q->next;

}

p = p->next;

}

return s3;

}

node \*delnode(node \*start, int data)

{

node \*tmp, \*p;

if (start == NULL)

{

printf("List is empty\n");

return start;

}

if (start->info == data)

{

tmp = start;

start = start->next;

free(tmp);

return start;

}

p = start;

while (p->next != NULL)

{

if (p->next->info == data)

{

tmp = p->next;

p->next = tmp->next;

free(tmp);

return start;

}

p = p->next;

}

printf("Element not found.\n");

return start;

}

node \*setdifference(node \*s1, node \*s2)

{

node \*s3 = NULL, \*temp;

while (s1 != NULL)

{

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

temp->info = s1->info;

temp->next = NULL;

if (s3 == NULL)

s3 = temp;

else

{

node \*q = s3;

while (q->next != NULL)

q = q->next;

q->next = temp;

}

s1 = s1->next;

}

while (s2 != NULL)

{

if (search(s3, s2->info))

{

delnode(s3, s2->info);

}

s2 = s2->next;

}

return s3;

}

int main()

{

node \*set1 = NULL, \*set2 = NULL;

int ch;

while (1)

{

printf("Enter 1 to enter elements of set 1 and 2.\n");

printf("Enter 2 to display elements of set 1 and 2.\n");

printf("Enter 3 for performing set union.\n");

printf("Enter 4 for performing set intersection.\n");

printf("Enter 5 for performing set difference.\n");

printf("Enter 6 to exit.\n");

scanf("%d", &ch);

switch (ch)

{

case 1:

set1 = NULL;

set2 = NULL;

set1 = create(set1, 1);

set2 = create(set2, 2);

break;

case 2:

display(set1, 1);

display(set2, 2);

break;

case 3:

display(setunion(set1, set2), 3);

break;

case 4:

display(setintersect(set1, set2), 3);

break;

case 5:

display(setdifference(set1, set2), 3);

break;

case 6:

exit(1);

default:

printf("Invalid input.\n");

}

}

}

**OUTPUT:-**

Enter 1 to enter elements of set 1 and 2.

Enter 2 to display elements of set 1 and 2.

Enter 3 for performing set union.

Enter 4 for performing set intersection.

Enter 5 for performing set difference.

Enter 6 to exit.

1

Set 1:

Enter the number of elements.

5

Enter the elements.

1 2 3 4 5

Set 2:

Enter the number of elements.

3

Enter the elements.

3 5 7

Enter 1 to enter elements of set 1 and 2.

Enter 2 to display elements of set 1 and 2.

Enter 3 for performing set union.

Enter 4 for performing set intersection.

Enter 5 for performing set difference.

Enter 6 to exit.

3

The elements of set 3 are : 1 2 3 4 5 7

Enter 1 to enter elements of set 1 and 2.

Enter 2 to display elements of set 1 and 2.

Enter 3 for performing set union.

Enter 4 for performing set intersection.

Enter 5 for performing set difference.

Enter 6 to exit.

4

The elements of set 3 are : 3 5

Enter 1 to enter elements of set 1 and 2.

Enter 2 to display elements of set 1 and 2.

Enter 3 for performing set union.

Enter 4 for performing set intersection.

Enter 5 for performing set difference.

Enter 6 to exit.

5

The elements of set 3 are : 1 2 4

Enter 1 to enter elements of set 1 and 2.

Enter 2 to display elements of set 1 and 2.

Enter 3 for performing set union.

Enter 4 for performing set intersection.

Enter 5 for performing set difference.

Enter 6 to exit.

2

The elements of set 1 are : 1 2 3 4 5

The elements of set 2 are : 3 5 7

Enter 1 to enter elements of set 1 and 2.

Enter 2 to display elements of set 1 and 2.

Enter 3 for performing set union.

Enter 4 for performing set intersection.

Enter 5 for performing set difference.

Enter 6 to exit.

6