```
#include <stdio.h>
#include <stdlib.h>

typedef struct poly {
    int coeff;
    int pow;
    struct poly *Next;
} Poly;

void Create(Poly *List);
void Display(Poly *List);
void Addition(Poly *Poly1, Poly *Poly2, Poly *Result);
void Subtraction(Poly *Poly1, Poly *Poly2, Poly *sub);
void Multiplication(Poly* Poly1, Poly* Poly2, Poly* multi);
void removeDuplicates(Poly* multi);
int main() {
    Poly *Poly1 = malloc(sizeof(Poly));
    Poly *Poly2 = malloc(sizeof(Poly));
    Poly *add = malloc(sizeof(Poly));
    Poly *sub = malloc(sizeof(Poly));
    Poly *multi = malloc(sizeof(Poly));
    Poly1->Next = NULL;
    Poly2->Next = NULL;

    printf("Enter the values for first polynomial:\n");
    Create(Poly1);

    printf("The polynomial equation is: ");
    Display(Poly1);
```

```
    printf("\nEnter the values for second polynomial:\n");
    Create(Poly2);

    printf("The polynomial equation is: ");
    Display(Poly2);
    int ch;
    do{
       printf("\n1.addition\n2.subraction\n3.multiplication\n");
       printf("Enter your choice:\n");
       scanf("%d",&ch);
       switch(ch){
                case 1:
                   Addition(Poly1,Poly2,add);
                   Display (add);
                case 2:
                   Subtraction(Poly1,Poly2,sub);
                   Display (sub);
                case 3:
                   Multiplication(Poly1,Poly2,multi);
                   Display (multi);
       }
    }while(ch<4);
}

 void Create(Poly *List) {
    int choice;
    Poly *Position, *NewNode;

    Position = List;
    do {
```

```
        NewNode = malloc(sizeof(Poly));
        printf("Enter the coefficient: ");
        scanf("%d", &NewNode->coeff);
        printf("Enter the power: ");
        scanf("%d", &NewNode->pow);
        NewNode->Next = NULL;
        Position->Next = NewNode;
        Position = NewNode;
        printf("Enter 1 to continue: ");
        scanf("%d", &choice);
    } while (choice == 1);
}

void Display(Poly *List) {
    Poly *Position;

    Position = List->Next;
    while (Position != NULL) {
        printf("%dx^%d", Position->coeff, Position->pow);
        Position = Position->Next;
        if (Position != NULL && Position->coeff > 0) {
            printf("+");
        }
    }
}

void Addition(Poly *Poly1, Poly *Poly2, Poly *add) {
    Poly *Position;
    Poly *NewNode;
```

```
Poly1 = Poly1->Next;
Poly2 = Poly2->Next;
add->Next = NULL;
Position = add;

while (Poly1 != NULL && Poly2 != NULL) {
    NewNode = malloc(sizeof(Poly));
    if (Poly1->pow == Poly2->pow) {
        NewNode->coeff = Poly1->coeff + Poly2->coeff;
        NewNode->pow = Poly1->pow;
        Poly1 = Poly1->Next;
        Poly2 = Poly2->Next;
    } else if (Poly1->pow > Poly2->pow) {
        NewNode->coeff = Poly1->coeff;
        NewNode->pow = Poly1->pow;
        Poly1 = Poly1->Next;
    } else if (Poly1->pow < Poly2->pow) {
        NewNode->coeff = Poly2->coeff;
        NewNode->pow = Poly2->pow;
        Poly2 = Poly2->Next;
    }
    NewNode->Next = NULL;
    Position->Next = NewNode;
    Position = NewNode;
}

while (Poly1 != NULL || Poly2 != NULL) {
    NewNode = malloc(sizeof(Poly));
    if (Poly1 != NULL) {
        NewNode->coeff = Poly1->coeff;
```

```
            NewNode->pow = Poly1->pow;
            Poly1 = Poly1->Next;
        }
        if (Poly2 != NULL) {
            NewNode->coeff = Poly2->coeff;
            NewNode->pow = Poly2->pow;
            Poly2 = Poly2->Next;
        }
        NewNode->Next = NULL;
        Position->Next = NewNode;
        Position = NewNode;
    }
}
void Subtraction(Poly *Poly1, Poly *Poly2, Poly *sub) {
    Poly *Position;
    Poly *NewNode;

    Poly1 = Poly1->Next;
    Poly2 = Poly2->Next;
    sub->Next = NULL;
    Position = sub;

    while (Poly1 != NULL && Poly2 != NULL) {
        NewNode = malloc(sizeof(Poly));
        if (Poly1->pow == Poly2->pow) {
            NewNode->coeff = Poly1->coeff - Poly2->coeff;
            NewNode->pow = Poly1->pow;
            Poly1 = Poly1->Next;
            Poly2 = Poly2->Next;
        } else if (Poly1->pow > Poly2->pow) {
```

```
            NewNode->coeff = Poly1->coeff;
            NewNode->pow = Poly1->pow;
            Poly1 = Poly1->Next;
        } else if (Poly1->pow < Poly2->pow) {
            NewNode->coeff = -(Poly2->coeff);
            NewNode->pow = Poly2->pow;
            Poly2 = Poly2->Next;
        }
        NewNode->Next = NULL;
        Position->Next = NewNode;
        Position = NewNode;
    }

    while (Poly1 != NULL || Poly2 != NULL) {
        NewNode = malloc(sizeof(Poly));
        if (Poly1 != NULL) {
            NewNode->coeff = Poly1->coeff;
            NewNode->pow = Poly1->pow;
            Poly1 = Poly1->Next;
        }
        if (Poly2 != NULL) {
            NewNode->coeff = -(Poly2->coeff);
            NewNode->pow = Poly2->pow;
            Poly2 = Poly2->Next;
        }
        NewNode->Next = NULL;
        Position->Next = NewNode;
        Position = NewNode;
    }
}
```

```
void removeDuplicates(Poly* multi)
{
    Poly *ptr1, *ptr2, *dup;
    ptr1 = multi;
    while (ptr1 != NULL && ptr1->Next != NULL) {
        ptr2 = ptr1;
        while (ptr2->Next != NULL) {
            if (ptr1->pow == ptr2->Next->pow) {
                ptr1->coeff = ptr1->coeff + ptr2->Next->coeff;
                dup = ptr2->Next;
                ptr2->Next = ptr2->Next->Next;
                free (dup);
            }
            else
                ptr2 = ptr2->Next;
        }
        ptr1 = ptr1->Next;
    }
}
void Multiplication(Poly* Poly1, Poly* Poly2, Poly* multi)
{
    Poly *NewNode, *Position;
    Poly1 = Poly1->Next;
    Poly2 = Poly2->Next;
    multi->Next = NULL;
    Position = multi;

    while (Poly1 != NULL) {
        while (Poly2 != NULL) {
            NewNode = malloc(sizeof(Poly));
```

```
            NewNode->coeff = Poly1->coeff * Poly2->coeff;
            NewNode->pow = Poly1->pow + Poly2->pow;
            Poly2 = Poly2->Next;
            NewNode->Next = NULL;
            Position->Next = NewNode;
            Position = NewNode;
        }
        Poly1 = Poly1->Next;
    }
}
```