Name: Yashwant Chandrakant Bhosale
MIS: 612303039
Div: SY Comp div 1

poly.h: Contains struct declarations and function prototypes for polynomial

```c
typedef struct {
    int c;
    int e;
} term;

typedef struct  {
    int deg;
    term *t;
} polynomial;

void init_p(polynomial *p, int n);
void append(polynomial *p, int c, int e);
void display(polynomial p);
polynomial *add(polynomial *p1, polynomial *p2);
polynomial *sub(polynomial *p1, polynomial *p2);
// helper functions
void sort(polynomial *p);
void swap(term *arr, int i, int j);
```

poly.c: Contains Function definitions for functions associated with polynomials

```c
#include <stdio.h>
#include <stdlib.h>
#include "poly.h"

// Function to initialize a polynomial
void init_p(polynomial *p, int n) {
    p->deg = n;
    p->t = (term *)malloc(n * sizeof(term));
    return;
}

// Function to append a term to a polynomial
void append(polynomial *p, int c, int e) {
    p->t[p->deg].c = c;
    p->t[p->deg].e = e;
    p->deg++;
    return;
}

// helper function: swap two terms in a polynomial
void swap(term *arr, int i, int j) {
    term temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
    return;
}
```

```
// Helper function: sort by degree
void sort(polynomial *p) {
    int i, j;
    for (i = 0; i < p->deg; i++) {
        for (j = i + 1; j < p->deg; j++) {
            if (p->t[i].e < p->t[j].e)
                swap(p->t, i, j);
        }
    }
    return;
}

// Function to display a polynomial
void display(polynomial p) {
    int i;
    sort(&p);
    for (i = 0; i < p.deg; i++) {
        printf("%dX^%d", p.t[i].c, p.t[i].e);
        if (i != p.deg - 1) {
            printf(" + ");
        }
    }
    printf("\n");
    return;
}
```

```c
// Function to add two polynomials
polynomial *add(polynomial *p1, polynomial *p2) {
    int i, j, k;
    polynomial *p3 = (polynomial *)malloc(sizeof(polynomial));
    init_p(p3, 0);
    i = j = k = 0;
    while (i < p1->deg && j < p2->deg) {
        // If the degree of the term in p1 is greater than the degree
of the term in p2, then append the term from p1 to p3
        if (p1->t[i].e > p2->t[j].e) {
            append(p3, p1->t[i].c, p1->t[i].e);
            i++;
        }
        // If the degree of the term in p2 is greater than the degree
of the term in p1, then append the term from p2 to p3
         else if (p1->t[i].e < p2->t[j].e) {
            append(p3, p2->t[j].c, p2->t[j].e);
            j++;
        }
        // If the degree of the terms in p1 and p2 is same then perform
addition and append term in p3
        else {
            append(p3, p1->t[i].c + p2->t[j].c, p1->t[i].e);
            i++;
            j++;
        }
    }
    // Append the remaining terms of p1 to p3
    while (i < p1->deg) {
        append(p3, p1->t[i].c, p1->t[i].e);
        i++;
    }
    // Append the remaining terms of p2 to p3
    while (j < p2->deg) {
        append(p3, p2->t[j].c, p2->t[j].e);
        j++;
    }
    return p3;
}
```

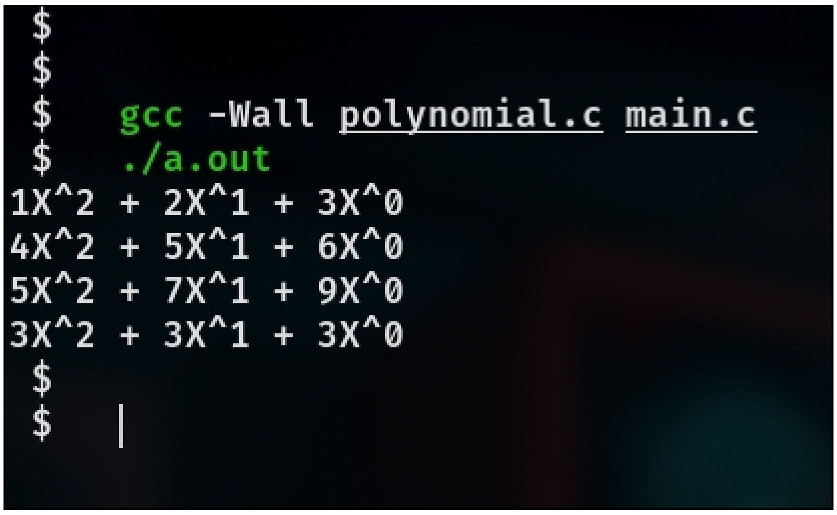
```
// Function to subtract two polynomials
polynomial *sub(polynomial *p1, polynomial *p2) {
    int i, j, k;
    polynomial *p3 = (polynomial *)malloc(sizeof(polynomial));
    init_p(p3, 0);
    i = j = k = 0;
    while (i < p1->deg && j < p2->deg) {
        // If the degree of the term in p1 is greater than the degree
of the term in p2, then append the term from p1 to p3
        if (p1->t[i].e > p2->t[j].e) {
            append(p3, p1->t[i].c, p1->t[i].e);
            i++;
        }
        // If the degree of the term in p2 is greater than the degree
of the term in p1, then append the term from p2 to p3
         else if (p1->t[i].e < p2->t[j].e) {
            append(p3, -p2->t[j].c, p2->t[j].e);
            j++;
        }
        // If the degree of the terms in p1 and p2 is same then perform
subtraction and append term in p3
        else {
            append(p3, p1->t[i].c - p2->t[j].c, p1->t[i].e);
            i++;
            j++;
        }
    }
    // Append the remaining terms of p1 to p3
    while (i < p1->deg) {
        append(p3, p1->t[i].c, p1->t[i].e);
        i++;
    }
    // Append the remaining terms of p2 to p3
    while (j < p2->deg) {
        append(p3, -p2->t[j].c, p2->t[j].e);
        j++;
    }
    return p3;
}
```

main.c: Contains Final execution of the program

```c
#include <stdio.h>
#include <stdlib.h>
#include "poly.h"

int main() {
    polynomial p1, p2, *p3, *p4;
    init_p(&p1, 0);
    init_p(&p2, 0);
    append(&p1, 1, 2);
    append(&p1, 2, 1);
    append(&p1, 3, 0);
    append(&p2, 4, 2);
    append(&p2, 5, 1);
    append(&p2, 6, 0);
    sort(&p1);
    sort(&p2);
    display(p1);
    display(p2);
    p3 = add(&p1, &p2);
    display(*p3);
    p4 = sub(&p2, &p1);
    display(*p4);
    return 0;
}
```

Output:



```
 $
 $
 $   gcc -Wall polynomial.c main.c
 $   ./a.out
1X^2 + 2X^1 + 3X^0
4X^2 + 5X^1 + 6X^0
5X^2 + 7X^1 + 9X^0
3X^2 + 3X^1 + 3X^0
 $
 $   |
```