```
#include <stdio.h>
#include <stdlib.h>
/*
Author:TB
Disclaimer: This work comes as is and with no claims or warranty
at all, implicit or explicit
            This is for educational purposes only.

  NOTE#1: This was written on a mac. You may need to adjust the
header files to suit
    your system.
    For a more conventional system comment out the headers above and
use the following
    headers instead.
    #include <stdio.h>
    #include <conio.h>
    #include <malloc.h>

    NOTE#2: This program assumes that the polynomials are entered in
proper order i.e.
    highest exponent first, then the immediately lesser etc. As an
added challenge, you may
    try to change that during your break if you so wish - this is
purely voluntary and not
    necessary at all.

    NOTE#3: Entering polynomials terminates when an exponent value
less than 0 is
    entered.
    */
     struct node
    {
              int num;
              int expo;
              struct node *next;
     };
     struct node *start1 = NULL;
     struct node *start2 = NULL;
     struct node *start3 = NULL;
     struct node *start4 = NULL;
     struct node *last3 = NULL;
     struct node *create_poly(struct node *);
     struct node *display_poly(struct node *, int polynum);
     struct node *add_poly(struct node *, struct node *, struct node
*);
     struct node *sub_poly(struct node *, struct node *, struct node
*);
     struct node *add_node(struct node *, int, int);
     void displayAll(struct node *, struct node *, struct node *,
struct node *);
```

```
int main()
{
    int option;
    do {
        printf("\n******* MAIN MENU *******");
        printf("\n 1. Enter the First polynomial");
        printf("\n 2. Display the First polynomial");
        printf("\n 3. Enter the second polynomial");
        printf("\n 4. Display the second polynomial");
        printf("\n 5. Add the polynomials");
        printf("\n 6. Display the result");
        printf("\n 7. Subtract the polynomials");
        printf("\n 8. Display the result");
        printf("\n 9. View all polynomials in system");
        printf("\n 10. EXIT");
        printf("\n\n Enter your option : ");
        scanf("%d", &option);
        printf(/*"\n You Entered  : %d\n\n", option*/ "\n\n");

        switch(option)
        {
            case 1: start1 = create_poly(start1);
            break;
            case 2: start1 = display_poly(start1,1);
            break;
            case 3: start2 = create_poly(start2);
            break;
            case 4: start2 = display_poly(start2, 2);
            break;
            case 5: start3 = add_poly(start1, start2, start3);
            break;
            case 6: start3 = display_poly(start3, 3);
            break;
            case 7: start4 = sub_poly(start1, start2, start4);
            break;
            case 8: start4 = display_poly(start4, 4);
            break;
            case 9: displayAll(start1, start2, start3, start4);
            break;
            case 10:
            break;
            default: printf("That is not one of the options!!\n");
            break;
        }

    }while(option!=10);
    printf("Bye!!\n");
    return 0;
}
```

```
struct node *create_poly(struct node *start)
{
    struct node *new_node, *ptr;
    int n, c;
    printf("\n Enter the coefficient : ");
    scanf("%d", &n);
    printf(" Enter its exponent : ");
    scanf("%d", &c);
    while(c >= 0)
    {
        if(start==NULL)
        {
            new_node = (struct node *)malloc(sizeof(struct node));
            new_node -> num = n;
            new_node -> expo = c;
            new_node -> next = NULL;
            start = new_node;
        } else {
            ptr = start;
            while(ptr -> next != NULL)
                ptr = ptr -> next;
            new_node = (struct node *)malloc(sizeof(struct node));
            new_node -> num = n;
            new_node -> expo = c;
            new_node -> next = NULL;
            ptr -> next = new_node;
        }
        printf("\n Enter the coefficient : ");
        scanf("%d", &n);
        if(c < 0)
            break;
        printf(" Enter its exponent : ");
        scanf("%d", &c);
    }
    return start;
}

struct node *display_poly(struct node *start, int polynum)
{
  struct node *ptr;
  ptr = start;
  if (polynum < 3)
      printf(" Poly #%d: ", polynum);
 else if (polynum == 3)
     printf("Poly Sum: ");
 else if (polynum == 4)
     printf("Poly Sub: ");
while(ptr != NULL)
{
```

```
        if (ptr->num>0)
        {
            printf("+");
        }
        printf("%dx%d", ptr -> num, ptr -> expo);
        ptr = ptr -> next;
    }
    printf("\n");
    return start;
}

struct node *add_node(struct node *start, int n, int c)
{
    struct node *ptr, *new_node;
    if(start == NULL) {
        new_node = (struct node *)malloc(sizeof(struct node));
        new_node -> num = n;
        new_node -> expo = c;
        new_node -> next = NULL;
        start = new_node;
    } else {
        ptr = start;
        while(ptr -> next != NULL)
            ptr = ptr -> next;
        new_node = (struct node *)malloc(sizeof(struct node));
        new_node -> num = n;
        new_node -> expo = c;
        new_node -> next = NULL;
        ptr -> next = new_node;
    }
    return start;
}

struct node *add_poly(struct node *start1, struct node *start2,
truct node *start3)
{
    struct node *ptr1, *ptr2;
    int sum_num, exp;
    ptr1 = start1, ptr2 = start2;
    while(ptr1 != NULL && ptr2 != NULL)
    {
        if(ptr1 -> expo == ptr2 -> expo)
        {
            sum_num = ptr1 -> num + ptr2 -> num;
            start3 = add_node(start3, sum_num, ptr1 -> expo);
            ptr1 = ptr1 -> next;
            ptr2 = ptr2 -> next;
        }
        else if(ptr1 -> expo > ptr2 -> expo)
        {
```

```
            start3 = add_node(start3, ptr1 -> num, ptr1 -> expo);
            ptr1 = ptr1 -> next;
        }
        else if(ptr1 -> expo < ptr2 -> expo)
        {
            start3 = add_node(start3, ptr2 -> num, ptr2 -> expo);
            ptr2 = ptr2 -> next;
        }
    }
    if(ptr1 == NULL)
    {
        while(ptr2 != NULL)
        {
            start3 = add_node(start3, ptr2 -> num, ptr2 -> expo);
            ptr2 = ptr2 -> next;
        }
    }
    if(ptr2 == NULL)
    {
        while(ptr1 != NULL)
        {
            start3 = add_node(start3, ptr1 -> num, ptr1 -> expo);
            ptr1 = ptr1 -> next;
        }
    }
    return start3;
}

struct node *sub_poly(struct node *start1, struct node *start2,
ct node *start4)
{
    struct node *ptr1, *ptr2;
    int sub_num;
    ptr1 = start1, ptr2 = start2;
    while(ptr1 != NULL && ptr2 != NULL)
    {
        if(ptr1 -> expo == ptr2 -> expo)
        {
            sub_num = ptr1 -> num - ptr2 -> num;
            start4 = add_node(start4, sub_num, ptr1 -> expo);
            ptr1 = ptr1 -> next;
            ptr2 = ptr2 -> next;
        }
        else if(ptr1 -> expo > ptr2 -> expo)
        {
            start4 = add_node(start4, ptr1 -> num, ptr1 -> expo);
            ptr1 = ptr1 -> next;
        }
        else if(ptr1 -> expo < ptr2 -> expo)
        {
```

```
            start4 = add_node(start4, -(ptr2 -> num), ptr2 ->
expo);
            ptr2 = ptr2 -> next;
        }
    }
    if(ptr1 == NULL)
    {
        while(ptr2 != NULL)
        {
            start4 = add_node(start4, -(ptr2 -> num), ptr2 ->
expo);
            ptr2 = ptr2 -> next;
        }
    }
    if(ptr2 == NULL)
    {
        while(ptr1 != NULL)
        {
            start4 = add_node(start4, ptr1 -> num, ptr1 -> expo);
            ptr1 = ptr1 -> next;
        }
    }
    return start4;
}

void displayAll(struct node *start1, struct node *start2, struct
node *start3, struct node *start4)
{
    start3 = add_poly(start1, start2, start3);
    start4 = sub_poly(start1, start2, start4);
    printf("\nPRINTING ALL POLYNOMIALS IN SYSTEM...\n");
    start1 = display_poly(start1,1);
    start2 = display_poly(start2,2);
    printf("-----------------------------------\n");
    start3 = display_poly(start3,3);
    start4 = display_poly(start4,4);
}
```