HackerRank For Work

MP8 B version 2 > micah_chuim@yahoo.com

Full Name:	Micah Nicole Chu Im
Email:	micah_chuim@yahoo.com
Test Name:	**MP8 B version 2**
Taken On:	4 May 2016 11:26:01 PHT
Time Taken:	31 min/ 90 min
Invited by:	Ryan
Invited on:	3 May 2016 18:29:10 PHT
Tags Score:	

Total Score
10/10

Candidate Feedback: (_____ _____) (; ;) (; ;) T.T T.T

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	**The Polynomials Have Many Faces** > **Coding**	19 min 19 sec	10/ 10	✓

QUESTION 1

Correct Answer

Score 10

The Polynomials Have Many Faces > Coding

QUESTION DESCRIPTION

The first line of input is the number of test cases, say *t*. *t* test cases will follow. Each test case is composed of 2 sets of numbers representing two polynomial expressions. Each polynomial expression is represented by a number *n* representing the number of terms present in the expression followed by *n* pairs of numbers representing the coefficient and exponent of each of the terms.

Represent the polynomial expressions using a list of terms.

Sample Input:
1
3
2 3 5 2 1 0
3
7 3 4 2 8 0

Sample output:
9x^3+9x^2+9
-5x^3+x^2-7

CANDIDATE ANSWER

Language used: **C**

```
#include <math.h>
```

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <assert.h>
#include <limits.h>
#include <stdbool.h>


typedef struct elem{
    int coef,exp; //data
    struct elem *next;
}node;

typedef struct{
    node *head, *tail;
    int size;
}list;


void init(list*);
void append(list*,int,int);
void display(list);
void display2(node*);
int deleteItem(list*,int);
node* itemAt(list,int);
int setItem(list*,int,int);
void simplify(list *l);
list add(list, list);
list sub(list,list);
void arrange(list *l);

int main() {

    list l1,l2,sum,diff;
    int t,n,x,y;
    int n2;

    init(&l1);
    init(&l2);

    scanf("%d",&t);

    do{

        scanf("%d",&n);

        do{

            scanf("%d %d",&x,&y);
            append(&l1,x,y);
         n--;
        }while(n>0);

         scanf("%d",&n2);

    do{

           scanf("%d %d",&x,&y);
            append(&l2,x,y);

        n2--;

    }while(n2>0);


    //display(l1);
    //simplify(&l1);
    //display(l1);
    //display(l2);
    //simplify(&l2);
    //display(l2);
    sum=add(l1,l2);
    //simplify(&sum);
    diff=sub(l1,l2);
    //display(sum);
```

```
    //display(sum);
    //display(diff);
    arrange(&sum);
    display(sum);
    arrange(&diff);
    display(diff);

    t--;

    init(&l1);
    init(&l2);

    }while(t>0);

    return 0;
}


void init(list* l){
    l->size = 0;
    l->head = l->tail = NULL;
}

void append(list* l,int x,int y){
    node *n = malloc(sizeof(node));

    n->coef = x;
    n->exp = y;
    n->next = NULL;
    if(l->size==0){
        l->head = l->tail = n;
    }
    else{
        l->tail->next = n;
        l->tail = n;
    }
    l->size++;
    }


void display(list l){
    node *tmp = l.head;
    while(tmp!=NULL){

        if(tmp->coef!=0){

        if(tmp->exp==0)
            printf("%d",tmp->coef);
        else if(tmp->exp==1)
            printf("%dx",tmp->coef);
        else if(tmp->coef==1)
            printf("x^%d",tmp->exp);
        else
            printf("%dx^%d",tmp->coef,tmp->exp);
    }
        tmp = tmp->next;


        if(tmp!=NULL){
            if(tmp->coef>0)
                printf("+");
        }
    }
    printf("\n");
}


int deleteItem(list* l,int pos){
    node *tmp = l->head, *del;
    int i = 1;
    if(pos<1 || pos>l->size)
        return 0;
    else{
        if(pos==1){
            del = l->head;
```

```
            l->head = del->next;
            del->next = NULL;
        }
        else{
            while(i<pos-1){
                tmp = tmp->next;
                i++;
            }
            del = tmp->next;
            tmp->next = del->next;
            del->next = NULL;
            if(del==l->tail)
                l->tail = tmp;
        }
        free(del);
        l->size--;
        return 1;
    }
}


node* itemAt(list l, int pos){
    node *tmp = l.head;
    int i = 1;

    while(i<pos){
        tmp = tmp->next;
        i++;
    }
    return tmp;
}

int setItem(list *l, int pos, int x){
    if(pos<1 || pos > l->size)
        return 0;

    node *tmp = l->head;
    int i = 1;

    while(i<pos){
        tmp = tmp->next;
        i++;
    }
    tmp->coef = x;
    return 1;
}

void simplify(list *l){
    int i=1,j;

    //display(*l);

    for(;i<l->size;i++){
        for(j=i+1;j<=l->size;j++){
            if(itemAt(*l,i)->exp==itemAt(*l,j)->exp){
              setItem(l,i,itemAt(*l,i)->coef + itemAt(*l,j)->coef );
                deleteItem(l,j);
                j--;
            }
        }
    }
}

list add(list l1, list l2){
    list sum;
    init(&sum);
    int c=1;

    while(c<=l1.size){
        append(&sum,itemAt(l1,c)->coef,itemAt(l1,c)->exp);
        c++;

    }

    c=1;
```

```
    while(c<=l2.size){
        append(&sum,itemAt(l2,c)->coef,itemAt(l2,c)->exp);
        c++;
    }

    //display(sum);
    simplify(&sum);
    return sum;
}

list sub(list l1, list l2){
    list sum;
    init(&sum);
    int c=1;

    while(c<=l1.size){
        append(&sum,itemAt(l1,c)->coef,itemAt(l1,c)->exp);
        c++;

    }

    c=1;
    while(c<=l2.size){
        append(&sum,(itemAt(l2,c)->coef)*-1,itemAt(l2,c)->exp);
        c++;
    }

    //display(sum);
    simplify(&sum);
    return sum;
}

void arrange(list *l){
    int i=1,j;
    int tmpcoef,tmpexp;
    //display(*l);

    for(;i<l->size;i++){
        for(j=i+1;j<=l->size;j++){
            if(itemAt(*l,i)->exp<=itemAt(*l,j)->exp){
                tmpcoef=itemAt(*l,j)->coef;
                tmpexp=itemAt(*l,j)->exp;
                itemAt(*l,j)->coef=itemAt(*l,i)->coef;
                itemAt(*l,j)->exp=itemAt(*l,i)->exp;
                itemAt(*l,i)->coef=tmpcoef;
                itemAt(*l,i)->exp=tmpexp;
            }
        }
    }
}

```

TESTCASE	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Success	3	0.0 sec	2.54 MB
Testcase 1	Easy	Success	5	0.0 sec	2.54 MB
Testcase 2	Easy	Success	2	0.0 sec	2.3 MB

No Comments