```c
#ifndef POLYNOME_H
#define POLYNOME_H

#include "asde-slist.h"


typedef SList polynome;

extern polynome polynome_creer();
extern void polynome_supprimer(polynome p);
extern void polynome_afficher(polynome p);
extern polynome polynome_modifier_monome(polynome p, int n, int a);
extern polynome polynome_additionner(polynome p1, polynome p2);
extern polynome polynome_soustraire(polynome p1, polynome p2);

#endif /* POLYNOME_H */
```



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

#include "polynome.h"


polynome
polynome_creer(){
  return NULL;
}

void
polynome_supprimer(polynome p){
  if(p != NULL){
    SList tmp;
    while(p != NULL){
      tmp= p;
      p= asde_slist_next(p);
      asde_slist_free_link(tmp);
    }
  }
}

polynome
polynome_modifier_monome(polynome p, int n, int a){
  if(n == 0)
    {
      p= asde_slist_delete_first(p);
      p= asde_slist_prepend(p,a);
    }
  else
    {
      int rang= 0;
      polynome p_crnt= p;
      polynome p_crnt_prev= NULL;
      while(p_crnt != NULL && rang < n){
        p_crnt_prev= p_crnt;
        p_crnt= asde_slist_next(p_crnt);
        rang++;
      }
      if((p_crnt == NULL) && (rang < n)){
        if( p_crnt_prev == NULL){
          p= asde_slist_prepend(p, 0);
          p_crnt_prev= p;
          rang++;
        }
        while(rang < n){
          p= asde_slist_insert_after(p, p_crnt_prev, 0);
          p_crnt_prev= asde_slist_next(p_crnt_prev);
          rang++;
        }
        p= asde_slist_insert_after(p, p_crnt_prev, a);
      }
      else
        {
          p= asde_slist_delete_after(p, p_crnt_prev);
          p= asde_slist_insert_after(p, p_crnt_prev, a);
        }
    }
  return p;
}
```

```
polynome
polynome_operation(polynome p1, polynome p2, char c){
  assert(c == '+'|| c == '-');
  polynome p= polynome_creer();
  while(p1 != NULL && p2 != NULL){
    int coeff1= asde_slist_data(p1);
    int coeff2= asde_slist_data(p2);
    int coeff= (c == '+')? coeff1 + coeff2 : coeff1 - coeff2;
    p= asde_slist_prepend(p, coeff);
    p1= asde_slist_next(p1);
    p2= asde_slist_next(p2);
  }
  while(p1 != NULL){
    p= asde_slist_prepend(p, asde_slist_data(p1));
    p1= asde_slist_next(p1);
  }

  while(p2 != NULL){
    p= asde_slist_prepend(p, asde_slist_data(p2));
    p2= asde_slist_next(p2);
  }
  polynome res= polynome_creer();
  while(p != NULL){
    res= asde_slist_prepend(res, asde_slist_data(p));
    p=asde_slist_next(p);
  }
  return res;
}

polynome
polynome_additionner(polynome p1, polynome p2){
  return polynome_operation(p1, p2, '+');
}
polynome
polynome_soustraire(polynome p1, polynome p2){
  return polynome_operation(p1, p2, '-');
}


void
polynome_afficher(polynome p){
  printf(" P(x)= ");
  if(p == NULL)
    printf(" 0 \n");
  else
    {

      int deg= 0;
      int coeff= asde_slist_data(p);
      if(coeff != 0)
        printf("\t %d * x^{%d}" ,coeff, deg);
      p= asde_slist_next(p);
      while(p != NULL){
        deg++;
        coeff= asde_slist_data(p);
        if(coeff != 0){
          if(coeff > 0)
            printf("\t + %d * x^{%d}" ,coeff, deg);
          else
            printf("\t %d * x^{%d}" ,coeff, deg);
        }
```



```
        p= asde_slist_next(p);
      }
    }
  printf("\n");
}
```

```c
#include <assert.h>
#include <stdlib.h>
#include <stdio.h>
#include "polynome.h"


int
main(void){
  polynome p= polynome_creer();
  p=polynome_modifier_monome(p, 2, -3);
  p=polynome_modifier_monome(p, 0, 1);
  p=polynome_modifier_monome(p, 1, 1);
  polynome_afficher(p);
  polynome q= polynome_creer();
  q=polynome_modifier_monome(q, 0, 2);
  q=polynome_modifier_monome(q, 1, -1);
  polynome_afficher(q);
  polynome r= polynome_additionner(p, q);
  polynome_afficher(r);
  polynome s= polynome_soustraire(p, q);
  polynome_afficher(s);
  return EXIT_SUCCESS;
}
```