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

#define N 10

typedef int* polynome;

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

polynome
polynome_creer(){
  polynome p= calloc(N, sizeof(int) * N);
  assert(p != NULL);
  return p;
}

void
polynome_supprimer(polynome p){
  if(p != NULL)
    free(p);
}


void
polynome_afficher(polynome p){
  printf(" P(x)= ");
  int i=0;
  while(p[i] == 0 && i<N){
    printf("\t");
    i++;
  }
  if(i == N)
    {
      printf(" 0 ");
    }
  else
    {
      if(p[i] < 0)
        printf("%d * x^{%d}", p[i], i);
      else
        printf(" %d * x^{%d}", p[i], i);
      for(int j=i+1; j<N; j++){
        if(p[j] == 0)
          printf("\t  ");
        else if (p[j] < 0)
          printf("  %d * x^{%d}", p[j], j);
        else
          printf(" + %d * x^{%d}", p[j], j);
      }
    }
  printf("\n");
}


void
polynome_modifier_monome(polynome p, int n, int a){
```



```c
  assert(n < N);
  p[n]= a;
}

polynome
polynome_additionner(polynome p1, polynome p2){
  polynome p= polynome_creer();
  assert(p != NULL);
  for(int i=0; i<N; i++)
    p[i]= p1[i] + p2[i];
  return p;
}

polynome
polynome_soustraire(polynome p1, polynome p2){
  polynome p= polynome_creer();
  assert(p != NULL);
  for(int i=0; i<N; i++)
    p[i]= p1[i] - p2[i];
  return p;
}



int
main(void){
  polynome p= polynome_creer();
  polynome_modifier_monome(p, 0, 1);
  polynome_modifier_monome(p, 1, 1);
  polynome_modifier_monome(p, 2, -3);
  polynome_afficher(p);
  polynome q= polynome_creer();
  polynome_modifier_monome(q, 0, 2);
  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;
}
```