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

void vec_write(short *v, int dim);
void vec_print(short *v, int dim);
void vec_clean(short *v, int dim);
void bin_sum(short *v1, short *v2, short *vR, short *vResult, int dim);
void bin_sub(short *v1, short *v2, short *vR, short *vResult, int dim);
void complement(short *v, short* vR, int dim);

int main () {
    //Inizio allocazione dinamica
    short *vA = (short*)calloc(32,sizeof(short));   //1st numero binario
    short *vB = (short*)calloc(32,sizeof(short));   //2nd numero binario
    short *vR = (short*)calloc(32,sizeof(short));   //Resto delle operazioni
    short *vResult = (short*)calloc(32,sizeof(short)); //Risultato
    char *ok = (char*)malloc(sizeof(char));         //Parametro di controllo
    //Fine allocazione dinamica

    vec_write(vA, 32);
    vec_print(vA, 32);
    printf("\n\n");
    vec_write(vB, 32);
    vec_print(vB, 32);

    printf("\n\nVuoi fare la somma? y/n: ");
    scanf("%s", ok);
    if (*ok == 'y' || *ok == 'Y') {
        printf("Eseguo la somma:\n");
        vec_clean(vR, 32); vec_clean(vResult, 32);  //Pulisco vettore del resto e del risultato
        bin_sum(vA, vB, vR, vResult, 32);       //Eseguo la somma
        vec_print(vResult, 32);                 //Stampo il risultato
    }

    printf("\n\nVuoi fare la sottrazione? y/n: ");
    scanf("%s", ok);
    if (*ok == 'y' || *ok == 'Y') {
        printf("Eseguo la sottrazione:\n");
        vec_clean(vR, 32); vec_clean(vResult, 32);  //Pulisco vettore del resto e del risultato
        bin_sub(vA, vB, vR, vResult, 32);       //Eseguo la sottrazione
        vec_print(vResult, 32);                 //Stampo il risultato
    }

    printf("\n\nVuoi fare la moltiplicazione? y/n: ");
    scanf("%s", ok);
    if (*ok == 'y' || *ok == 'Y') {
        printf("Eseguo la moltiplicazione:\n");
    }
    free(vA); free(vB); free(vR); free(vResult); free(ok);  //Libero la memoria allocata dinamicamente
    return 0;
}

void vec_write(short *v, int dim){
    int a, mask;
    printf("Inserisci il numero decimale\t");
    scanf("%d", &a);

    for (int i=(dim-1); i>=0; i--) {
        mask = 1<<i;
        if ((mask&a) == 0) v[i] = 0;
        else v[i] = 1;
    }
}
```

```c

void vec_print(short *v, int dim) {
    for (int i=(dim-1); i>=0; i--) {
        printf("%hd", v[i]);
    }
}

void vec_clean(short *v, int dim) {
    for (int i=0; i<dim; i++) {
        v[i] = 0;
    }
}

void bin sum(short *v1, short *v2, short *vR, short *vResult, int dim) {
    for (int i=0; i<dim; i++) {
        if (vR[i] == 0) {
            //printf("Resto 0  ");
            if (v1[i] == v2[i] && v1[i] == 0) {
                //printf("%d == %d, next resto 0\n", v1[i], v2[i]);
                vR[i+1] = 0;
                vResult[i] = 0;
            }else if (v1[i] != v2[i]) {
                //printf("%d != %d, next resto 0\n", v1[i], v2[i]);
                vR[i+1] = 0;
                vResult[i] = 1;
            }else if (v1[i] == v2[i] && v1[i] != 0) {
                //printf("%d == %d, next resto 1\n", v1[i], v2[i]);
                vR[i+1] = 1;
                vResult[i] = 0;
            }
        }else {
            //printf("Resto 1  ");
            if (v1[i] == v2[i] && v1[i] == 0) {
                //printf("%d == %d, next resto 0\n", v1[i], v2[i]);
                vR[i+1] = 0;
                vResult[i] = 1;
            }else if (v1[i] != v2[i]) {
                //printf("%d != %d, next resto 1\n", v1[i], v2[i]);
                vR[i+1] = 1;
                vResult[i] = 0;
            }else if (v1[i] == v2[i] && v1[i] != 0) {
                //printf("%d == %d, next resto 1\n", v1[i], v2[i]);
                vR[i+1] = 1;
                vResult[i] = 1;
            }
        }
    }
}

void complement(short *v, short* vR, int dim) {
    for (int i=0; i<dim; i++) {
        if (v[i] == 0) v[i]=1;
        else v[i] = 0;
    }
    short *one = (short*)calloc(32, sizeof(short));
    vec_clean(one, 32);
    one[0] = 1;
    bin sum(v, one, vR, v, 32);
    vec clean(vR, 32); free(one);
}

void bin sub(short *v1, short *v2, short *vR, short *vResult, int dim) {
    complement(v2, vR, 32);
    bin_sum(v1, v2, vR, vResult, 32);
}

```