

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void vec write(short *v, int dim);
5  void vec_print(short *v, int dim);
6  void vec clean(short *v, int dim);
7  void bin sum(short *v1, short *v2, short *vR, short *vResult, int dim);
8  void bin_sub(short *v1, short *v2, short *vR, short *vResult, int dim);
9  void complement(short *v, short* vR, int dim);
10
11 int main () {
12     //Inizio allocazione dinamica
13     short *vA = (short*)calloc(32,sizeof(short)); //1st numero binario
14     short *vB = (short*)calloc(32,sizeof(short)); //2nd numero binario
15     short *vR = (short*)calloc(32,sizeof(short)); //Resto delle operazioni
16     short *vResult = (short*)calloc(32,sizeof(short)); //Risultato
17     char *ok = (char*)malloc(sizeof(char)); //Parametro di controllo
18     //Fine allocazione dinamica
19
20     vec_write(vA, 32);
21     vec_print(vA, 32);
22     printf("\n\n");
23     vec_write(vB, 32);
24     vec_print(vB, 32);
25
26     printf("\n\nVuoi fare la somma? y/n: ");
27     scanf("%s", ok);
28     if (*ok == 'y' || *ok == 'Y') {
29         printf("Eseguo la somma:\n");
30         vec_clean(vR, 32); vec_clean(vResult, 32); //Pulisco vettore del resto e
           del risultato
31         bin sum(vA, vB, vR, vResult, 32); //Eseguo la somma
32         vec_print(vResult, 32); //Stampo il risultato
33     }
34
35     printf("\n\nVuoi fare la sottrazione? y/n: ");
36     scanf("%s", ok);
37     if (*ok == 'y' || *ok == 'Y') {
38         printf("Eseguo la sottrazione:\n");
39         vec_clean(vR, 32); vec_clean(vResult, 32); //Pulisco vettore del resto e
           del risultato
40         bin_sub(vA, vB, vR, vResult, 32); //Eseguo la sottrazione
41         vec_print(vResult, 32); //Stampo il risultato
42     }
43
44     printf("\n\nVuoi fare la moltiplicazione? y/n: ");
45     scanf("%s", ok);
46     if (*ok == 'y' || *ok == 'Y') {
47         printf("Eseguo la moltiplicazione:\n");
48     }
49     free(vA); free(vB); free(vR); free(vResult); free(ok); //Libero la memoria
           allocata dinamicamente
50     return 0;
51 }
52
53 void vec_write(short *v, int dim){
54     int a, mask;
55     printf("Inserisci il numero decimale\t");
56     scanf("%d", &a);
57
58     for (int i=(dim-1); i>=0; i--) {
59         mask = 1<<i;
60         if ((mask&a) == 0) v[i] = 0;
61         else v[i] = 1;
62     }
63 }

```

```

64
65 void vec_print(short *v, int dim) {
66     for (int i=(dim-1); i>=0; i--) {
67         printf("%hd", v[i]);
68     }
69 }
70
71 void vec_clean(short *v, int dim) {
72     for (int i=0; i<dim; i++) {
73         v[i] = 0;
74     }
75 }
76
77 void bin_sum(short *v1, short *v2, short *vR, short *vResult, int dim) {
78     for (int i=0; i<dim; i++) {
79         if (vR[i] == 0) {
80             //printf("Resto 0 ");
81             if (v1[i] == v2[i] && v1[i] == 0) {
82                 //printf("%d == %d, next resto 0\n", v1[i], v2[i]);
83                 vR[i+1] = 0;
84                 vResult[i] = 0;
85             } else if (v1[i] != v2[i]) {
86                 //printf("%d != %d, next resto 0\n", v1[i], v2[i]);
87                 vR[i+1] = 0;
88                 vResult[i] = 1;
89             } else if (v1[i] == v2[i] && v1[i] != 0) {
90                 //printf("%d == %d, next resto 1\n", v1[i], v2[i]);
91                 vR[i+1] = 1;
92                 vResult[i] = 0;
93             }
94         } else {
95             //printf("Resto 1 ");
96             if (v1[i] == v2[i] && v1[i] == 0) {
97                 //printf("%d == %d, next resto 0\n", v1[i], v2[i]);
98                 vR[i+1] = 0;
99                 vResult[i] = 1;
100             } else if (v1[i] != v2[i]) {
101                 //printf("%d != %d, next resto 1\n", v1[i], v2[i]);
102                 vR[i+1] = 1;
103                 vResult[i] = 0;
104             } else if (v1[i] == v2[i] && v1[i] != 0) {
105                 //printf("%d == %d, next resto 1\n", v1[i], v2[i]);
106                 vR[i+1] = 1;
107                 vResult[i] = 1;
108             }
109         }
110     }
111 }
112
113 void complement(short *v, short* vR, int dim) {
114     for (int i=0; i<dim; i++) {
115         if (v[i] == 0) v[i]=1;
116         else v[i] = 0;
117     }
118     short *one = (short*)calloc(32, sizeof(short));
119     vec_clean(one, 32);
120     one[0] = 1;
121     bin_sum(v, one, vR, v, 32);
122     vec_clean(vR, 32); free(one);
123 }
124
125 void bin_sub(short *v1, short *v2, short *vR, short *vResult, int dim) {
126     complement(v2, vR, 32);
127     bin_sum(v1, v2, vR, vResult, 32);
128 }
129

```