UNIVERSITY OF TRENTO - Italy
Department of Information
and Communication Technology

Soluzioni Esercizi Teorici Sessioni Esami

DISI – aa 2024/25

Pierluigi Roberti
Carmelo Ferrante

Università degli Studi di Trento

010920 Prova PROG1

[Teo02-3punti]
[TEO] Dire quale numero intero è rappresentato dalla seguente sequenza 10011_2
Nel caso in cui il numero venga considerato come:
a) numero intero senza segno
b) numero intero con segno
c) numero in complemento a 2 (CA2)

mostrare tutti i passaggi utilizzati per ottenere la soluzione.

Soluzione

sequenza 10011_2

a) numero intero senza segno

$1*2^4$	+	$0*2^3$	+	$0*2^2$	+	$1*2^1$	+	$1*2^0$	=
16	+	0	+	0	+	2	+	1	=

= 19

b) numero intero con segno

1=-		$0*2^3$	+	$0*2^2$	+	$1*2^1$	+	$1*2^0$	=
-		0	+	0	+	2	+	1	=

= -3

c) numero in complemento a 2 (CA2)

1=-		0		0		1		1	
1=-		1		1		0		1	
1=-		$1*2^3$	+	$1*2^2$	+	$0*2^1$	+	$1*2^0$	=
-		8	+	4	+	0	+	1	=

= -13

Si invertono tutti i bit e si aggiunge 1.

In breve: partendo da destra, copio bit a bit fino al primo 1, dopo inverto i bit

Eccezione: non valido per estremo inferiore rispetto al n. di bit
es. 1000 0000 = -128

Altri esempi CA2

$00010110 = 2^1 + 2^2 + 2^4 = 22$

$11100101100 = (-)0011010100 = 2^2 + 2^4 + 2^6 + 2^7 = -212$

$10101101101 = (-)\ 1010010011 = 2^0 + 2^1 + 2^4 + 2^7 + 2^9 = -659$

$11010111 = (-)\ 0101001 = 2^0 + 2^3 + 2^5 = -41$

$00110011 = 2^0 + 2^1 + 2^4 + 2^5 = 51$

$10000000 = (-)\ 10000000 = -128$

$11111111 = (-)\ 0000001 = 2^0 = -1$

$00000000 = 0$

$01111111 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 127.$

070220 Prova PROG1

[Domanda 1 – punti 2]

Nell'uso dei puntatori in C++ la scrittura "->" vuol dire:

SI NO riferimento

SI NO dereferenziazione e accesso al campo

SI NO accesso al campo e dereferenziazione

SI NO solo deferenziazione

Soluzione

& è l'operatore unario di referenziazione. La sua operazione complementare è la dereferenziazione o indirezione, effettuabile con l'operatore unario *

Gli operatori binari . e -> svolgono la stessa funzione quando l'operando sinistro è una struttura (o simile) e servono ad accedere all'operando destro che deve essere un membro dell'operando sinistro.

[Domanda 1 – punti 2]

Nell'uso dei puntatori in C++ la scrittura "->" vuol dire:

SI NO riferimento
SI NO dereferenziazione e accesso al campo
SI NO accesso al campo e dereferenziazione
SI NO solo deferenziazione

070220 Prova PROG1

Se inseriamo un numero dove c'è una condizione (ad es. WHILE (1)) come viene valutato?

SI NO Come una condizione falsa se il numero è diverso da zero
SI NO Come una condizione falsa se il numero è minore di zero
SI NO Come una condizione falsa se il numero è zero
SI NO Come una condizione vera solo se il numero è uno

Soluzione

Ricordiamo che nel C non esiste il tipo di variabile booleana e le condizioni vengono verificato con l'utilizzo dei numeri interi

```
int controllo = 0;
if (controllo) {
   //condizione non verificata
} else {
   //condizione verificata
}

int controllo = 1;
if (controllo) {
   // condizione verificata
} else {
   // condizione non verificata
}
```

Allo stesso modo

`while(1)` implica una condizione sempre vera

`while(0)` implica una condizione sempre falsa

Soluzione

Se inseriamo un numero dove c'è una condizione (ad es. WHILE (1)) come viene valutato?

SI NO Come una condizione falsa se il numero è diverso da zero
SI NO Come una condizione falsa se il numero è minore di zero
SI NO Come una condizione falsa se il numero è zero
SI NO Come una condizione vera solo se il numero è uno

070220 Prova PROG1

Possiamo usare una variabile dichiarata in un ciclo FOR (ad es. la variabile k in FOR (int k = 0; k < 10; k++)…) al di fuori di tale ciclo?

SI NO no

SI NO sì

SI NO sì se il FOR è nel codice di una funzione

SI NO sì, ma solo se c'è una sola istruzione senza { }

Soluzione

Vi sono **quattro** regole di visibilità delle variabili, dette anche regole di campo d'azione. Sono:

- il blocco;
- la funzione;
- il file;
- il programma.

Una variabile dichiarata in un blocco può essere utilizzata solo all'interno di quel blocco.

Una variabile dichiarata in una funzione può essere utilizzata solo all'interno di quella funzione.

Una variabile dichiarata al di fuori di una funzione può essere utilizzata in qualsiasi punto di quello stesso file successivo al punto in cui è stata dichiarata.

Una variabile dichiarata come **extern** può essere usata in ogni file del programma.

Possiamo usare una variabile dichiarata in un ciclo FOR (ad es. la variabile k in FOR (int k = 0; k < 10; k++)…) al di fuori di tale ciclo?

SI NO no

SI NO sì

SI NO sì se il FOR è nel codice di una funzione

SI NO sì, ma solo se c'è una sola istruzione senza { }

070220 Prova PROG1

[Domanda 2 – punti 2]

Quanti bit sono necessari per rappresentare tutti i numeri tra -20_{10} e + 20_{10} (estremi inclusi) in complemento a due? (dettagliare le risposte)

Codificare il numero -12_{10} in complemento a 2

Soluzione

Avendo n bit si possono rappresentare tutti i numeri interi fino a 2^n-1

Nel caso, però, di complemento a due o di signed integer, un bit viene utilizzato per il solo segno, quindi in complemento a 2, n bit possono rappresentare tutti i numeri da -2^{n-1} fino a $2^{n-1}-1$

Quindi con 3 bit è possibile rappresentare i numeri da -2^2 a 2^2-1 = -4 a +3

Con 6 bit è possibile rappresentare i numeri da -2^5 a 2^5-1 = -32 a +31

Servono quindi 6 bit, di cui uno per il segno e i rimanenti per i numeri da -32 a +31

Per trasformare -12 in complemento a 2 partiamo dal numero 12 in binario =
1100

E quindi trasformiamolo aggiungendo un bit per il segno in cima, invertendo tutti i bit e quindi aggiungendo 1

10011 + 1 = 10100

100120 Prova PROG1

[Domanda 1 – punti 2] Calcolare la somma tra $(75)_8$ e $(22)_8$.

Inoltre dire a quali delle soluzioni equivale il risultato, mostrando tutti i passaggi per la somma

e le conversioni

SI NO $(117)_8$

SI NO $(100111)_2$

SI NO $(6F)_{16}$

SI NO $(79)_{10}$

Soluzione

Calcolo la somma in base 8

Metodo 1: sommo direttamente $(75)_8$ e $(22)_8$

```
   75 +
   22 =
--------------
  (117)8
```

Metodo 2: converto in base 10, dopo sommo e riconverto

$75 = 7*8^1 + 5*8^0 = 56 + 5 = 61$
$22 = 2*8^1 + 2*8^0 = 16 + 2 = 18$

$61 + 18 = 79$

Converto in base 8:
79 % 8 =

9 resto	7
1 resto	1
0 resto	1

Soluzione

Converto la somma nelle altre basi (partendo dalla base 10 per comodità)

Base 10

$117_8 = 1*8^2 + 1*8^1 + 7*8^0 = 64 + 8 + 7 = 79_{10}$

Base 2

79 % 2

39 resto 1
19 resto 1
9 resto 1
4 resto 1
2 resto 0
1 resto 0
0 resto 1

1001111

Base 16

79 % 16

4 resto 15
0 resto 4

4F

Soluzione

[Domanda 1 – punti 2] Calcolare la somma tra $(75)_8$ e $(22)_8$.

Inoltre dire a quali delle soluzioni equivale il risultato, mostrando tutti i passaggi per la somma

e le conversioni

SI NO $(117)_8$

SI NO $(100111)_2$

SI NO $(6F)_{16}$

SI NO $(79)_{10}$

270720 Prova PROG1

[Teo02-3punti]

[TEO] Dire quale è il risultato della somma tra 14_5 e 23_6 ?

Mostrare in dettaglio tutti i passaggi e scrivere il risultato della somma in base 2, in base 8 ed in base 16

Soluzione

Uniformiamo le basi a 10

$14_5 = 1*5^1 + 4*5^0 = 5 + 4 = 9$

$23_6 = 2*6^1 + 3*6^0 = 12 + 3 = 15$

Sommiamo 9 + 15 = 24

Convertiamolo in base 2, 8 e 16

Base 2

24 % 2 =

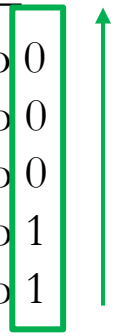
12 resto 0

6 resto 0

3 resto 0

1 resto 1

0 resto 1



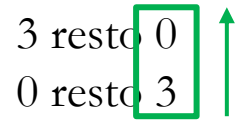
11000

Base 8

24 % 8 =

3 resto 0

0 resto 3



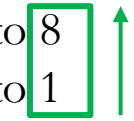
30

Base 16

24 % 16 =

1 resto 8

0 resto 1



18

Soluzione

Metodo 2: conversione da binario a base 8 e 16

11000

In base 8: raggruppo a 3 a 3 = 011 000 = 30

In base 16: raggruppo a 4 a 4 = 0001 1000 = 18

Base 2

24 % 2 =

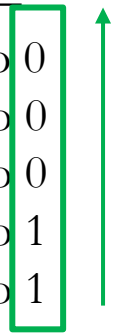
12 resto 0

6 resto 0

3 resto 0

1 resto 1

0 resto 1



11000

300620 Prova PROG1

[Teo02-3punti]
Dire quale è il risultato della somma tra 14_8 e 16_{16}?
Scrivere il risultato in base 2, in base 8 ed in base 16

Soluzione

Uniformiamo le basi a 10

$14_8 = 1*8^1 + 4*8^0 = 8 + 4 = 12$

$16_{16} = 1*16^1 + 3*16^0 = 16 + 3 = 19$

Sommiamo 12 + 19 = 31

Convertiamolo in base 2, 8 e 16

Base 2

31 % 2 =

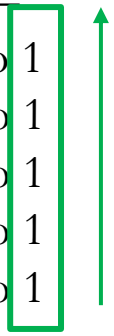
15 resto 1

7 resto 1

3 resto 1

1 resto 1

0 resto 1



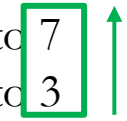
11111

Base 8

31 % 8 =

3 resto 7

0 resto 3



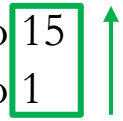
37

Base 8

31 % 16 =

1 resto 15

0 resto 1



1F

300620 Prova PROG1

[Teo05-3punti]
Considerare la dichiarazione float dati[10];
Dato il codice seguente:
dati[2]=5;
2[dati]=5;
*(dati+2)=5;
spiegare in dettaglio se le istruzioni sono o non sono equivalenti dettagliando la risposta.

Soluzione

dati[2]=5;
Assegna il valore 5 al 3° elemento del vettore dati

*(dati+2)=5;
Prende l'indirizzo di dati, aggiunge 2 e vi accede per dereferenziazione

2[dati]=5;
Accedere all'elemento di un array usando la forma ptr[3], equivale ad una forma breve per scrivere *(ptr + 3). Poiché quest'ultimo è equivalente a *(3+ptr), ne deriva che anche 3[ptr] è una forma valida.
Anche nel nostro caso, quindi, 2[dati] = 5 è una forma valida

Quindi
dati[2]=5;
È equivalente a
*(dati+2)=5;
È eqivalente a
2[dati]=5;

040919 Prova PROG1

[Domanda 4 - punti 4] Data la seguente struttura:

```
typedef struct Tnodo{
        int dato;
        Tnodo *next;
} Tnodo;
```

scrivere una funzione che verificare che la lista contenga dati ordinati in modo decrescente.

Definire e creare una funzione a cui viene passato in modo opportuno la struttura e che restituisce 1 se è ordinato e 0 se non lo è.

Soluzione

```
int verificaDecrescente(Tnodo* s) {
        if (s==NULL){
                return 1;
        }
        int last = s->dato;
        s = s->next;
        while (s != NULL) {
                if (last < s->dato) {
                        return 0;
                }
                last = s->dato;
                s = s->next;
        }
        return 1;
}
```

Codice di test

```
#include <iostream>
using namespace std;
typedef struct Tnodo{
	int dato;
	Tnodo *next;
	Tnodo(int d) {	dato = d; next = NULL; }
} Tnodo;
int verificaDecrescente(Tnodo* s) {
	if (s==NULL){ return 1; }
	int last = s->dato;
	s = s->next;
	while (s != NULL) {
		if (last < s->dato) { return 0; }
		last = s->dato;
		s = s->next;
	}
	return 1;
}
int main(){
	Tnodo* nodo = new Tnodo(5);
	nodo->next = new Tnodo(3);
	nodo->next->next = new Tnodo(0);
	cout << "Lista 1 decrescente: " << verificaDecrescente(nodo) << endl;
	Tnodo* nodo2 = new Tnodo(5);
	nodo2->next = new Tnodo(3);
	nodo2->next->next = new Tnodo(4);
	cout << "Lista 2 non decrescente: " << verificaDecrescente(nodo2) << endl;
}
```

060218 Prova PROG1

[Domanda 1 - punti 4] Dato un file di testo "persone.txt", si supponga che sia costituito da righe ciascuna contenente una stringa (nome della persona) ed un intero (età). Ad esempio:

mario 31

lucia 28

gianni 22

….

Scrivere la porzione di codice in grado di i) aprire il file, ii) leggere il contenuto del file e iii) stampare a video i nomi delle persone che hanno l'età maggiore e minore.

Nota. Nel caso di valori multipli di anni massimi o minimi, considerare il primo.

Codice di test

```
#include <iostream>
#include <cstring>
#define MAX 30
using namespace std;
int main(){
	FILE *nomi;
	nomi = fopen("persone.txt", "r");
	if(nomi==NULL){
		printf("Errore apertura file dati.txt");
		return 1;
	}
	char nome[MAX] = "\0";
	int eta = 0, minimo = 0, massimo = 0;
	char nome_minimo[MAX] = "\0", nome_massimo[MAX] = "\0";
	while( fscanf(nomi, "%s %d", &nome, &eta) == 2 ){
		if (eta > massimo || massimo == 0) {
				massimo = eta;  strcpy(nome_massimo, nome);
		}
		if (eta < minimo || minimo == 0) {
				minimo = eta; strcpy(nome_minimo, nome);
		}
	}
	fclose(nomi);

	cout << "Massimo: " << nome_massimo << " " << massimo << endl;
	cout << "Minimo: " << nome_minimo << " " << minimo << endl;
}
```

060218 Prova PROG1

[Domanda 3 - punti 4] Descrivere un algoritmo che moltiplichi due interi, n, m, senza usare la moltiplicazione o divisione e che permetta di riempire la tabella di sotto.

Qual è il migliore algoritmo che potete progettare, ovvero che abbia T(n) (funzione delle istruzioni da eseguire) migliore.

n,m	1	2	3	4	5
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25

Soluzione

```
#include <iostream>
#include <cstring>
#define MAX 30
using namespace std;
int main(){

	int n, m;

	scanf("%d %d", &n, &m);

	int somma = 0;
	for (int i=0; i<m; i++) {
		somma += n;
	}

	cout << n << "x" << m << " = " << somma;

}
```

Soluzione efficiente

```
#include <iostream>
using namespace std;
int moltiplica(int n, int m) {
	int somma = 0;   for (int i=0; i<m; i++) { somma += n; } return somma;
}
void stampa(int matrice[6][6]) {
	for (int n=1; n<=5; n++) {
		for (int m=1; m<=5; m++) {
				if (matrice[n][m] < 10) { cout << " "; }
				cout << matrice[n][m] << " ";
		}
		cout << endl;
	}
}
int main(){
	int n, m, operazioni = 0;
	int matrice[6][6];
	for (int n=1; n<=5; n++) {
		for (int m=n; m<=5; m++) { matrice[n][m] = matrice[m][n] = moltiplica(n, m);
	operazioni++; }
	}
	cout << "N. Operazioni: " << operazioni << endl;
	stampa(matrice);
}
```

3° Appello - Prova scritta – 06/06/2016

[Domanda 9 - punti 3] Dire a quale dichiarazione si deve fare riferimento nelle diverse istruzioni che coinvolgono la x. Scrivere inoltre cosa viene stampato da programma.

```
int x=0; /* Dichiarazione A*/
funz1() {
        x = x + 3;                  // A ☐ B ☐ C ☐
        cout<< “funz1:” << x;       // A ☐ B ☐ C ☐
        int x = 5; /* Dichiarazione C*/
}
funz2() {
        int x = 10; /* Dichiarazione B*/
        funz1();
        cout<< “funz2:” << x;       // A ☐ B ☐ C ☐
}
main() {
        x++;                        // A ☐ B ☐ C ☐
        cout <<“main:” << x;        // A ☐ B ☐ C ☐
        funz2();
        cout <<“main:” << x;        // A ☐ B ☐ C ☐
}
```

3° Appello - Prova scritta – 06/06/2016

[Domanda 9 - punti 3] Dire a quale dichiarazione si deve fare riferimento nelle diverse istruzioni che coinvolgono la x. Scrivere inoltre cosa viene stampato da programma.

```
int x=0; /* Dichiarazione A*/
funz1() {
        x = x + 3;                  // A [X] B [ ] C [ ]
        cout<< “funz1:” << x;       // A [X] B [ ] C [ ]
        int x = 5; /* Dichiarazione C*/
}
funz2() {
        int x = 10; /* Dichiarazione B*/
        funz1();
        cout<< “funz2:” << x;       // A [ ] B [X] C [ ]
}
main() {
        x++;                        // A [X] B [ ] C [ ]
        cout <<“main:” << x;        // A [X] B [ ] C [ ]
        funz2();
        cout <<“main:” << x;        // A [X] B [ ] C [ ]
}
```

main:1
funz1:4
funz2:10
main:4

5° Appello - Prova scritta – 11/09/2015

[Domanda 2 – punti 4] Cosa contengono le variabili x e y dopo l'esecuzione delle seguenti istruzioni:

```
int x = 4, y = 7;
x = y + (y = ++x);
```
x [ ] y [ ]

```
int x = 3, y = 6;
x = y + (x = x--);
```
x [ ] y [ ]

```
int x = 2, y = 5;
y = (x = y++) +x;
```
x [ ] y [ ]

```
int xy (int x, int y){
  return (x==y) ? x : y ;
}
int x = 3, y = 3;
y += xy (y, x);
```
x [ ] y [ ]

5° Appello - Prova scritta – 11/09/2015

[Domanda 2 – punti 4] Cosa contengono le variabili x e y dopo l'esecuzione delle seguenti istruzioni:

```
int x = 4, y = 7;
x = y + (y = ++x);
```
x [10] y [5]

```
int x = 3, y = 6;
x = y + (x = x--);
```
x [9] y [6]

```
int x = 2, y = 5;
y = (x = y++) +x;
```
x [5] y [10]

```
int xy (int x, int y){
  return (x==y) ? x : y ;
}
int x = 3, y = 3;
y += xy (y, x);
```
x [3] y [6]

3° Appello - Prova scritta – 10/06/2014

Cosa contengono x, y dopo l'esecuzione delle seguenti istruzioni?

```
int x=3, y=1;
if (x<=y && (x=y++)>0)
{ x = y;}
```

x [ ] y [ ]

Cosa contengono x, y dopo l'esecuzione delle seguenti istruzioni?

```
int x=4, y=-1;
while (x != y) {
 x = x + y;
 y = y ;
}
```

x [ ] y [ ]

Cosa contengono a, b dopo l'esecuzione delle seguenti istruzioni?

```
function non(b,a)
{a=3;return a+b;}
int a=1, b=4
b = non(a,b);
```

a [ ] b [ ]

Cosa contengono x, y dopo l'esecuzione delle seguenti istruzioni?

```
int x=1;   int y=1;
if (x == y){
      do {y--;
      } while (x >= y)
}
```

x [ ] y [ ]

3° Appello - Prova scritta – 10/06/2014

Cosa contengono x, y dopo l'esecuzione delle seguenti istruzioni?

```
int x=3, y=1;
if (x<=y && (x=y++)>0)
{ x = y;}
```

x [3] y [1]

Cosa contengono x, y dopo l'esecuzione delle seguenti istruzioni?

```
int x=4, y=-1;
while (x != y) {
 x = x + y;
 y = y ;
}
```

x [-1] y [-1]

Cosa contengono a, b dopo l'esecuzione delle seguenti istruzioni?

```
function non(b,a)
{a=3;return a+b;}
int a=1, b=4
b = non(a,b);
```

a [1] b [4]

Cosa contengono x, y dopo l'esecuzione delle seguenti istruzioni?

```
int x=1;   int y=1;
if (x == y){
      do {y--;
      } while (x >= y)
}
```

x [1] y [-MAX]

3° Appello - Prova scritta – 18/06/2019

```
int calcola (int *a, int b, int *c){
  *a = *a * b;          printf("a=%d\n",*a);
  b = b * 2;            printf("b=%d\n",b);
  *c = *a – b;          printf("c=%d\n",*c);
  return *a + b + *c;
}

int main (){
  int x, z;
  int *y;
  y = &x;
  z = -2;
  *y = 3;
  x = 2;
  printf("x=%d y=%d z=%d\n",x,*y,z);
  z = calcola(y,x,&z);
  printf("x=%d y=%d z=%d\n",x,*y,z);
 }
```

cosa viene stampato dall'esecuzione del programma

3° Appello - Prova scritta – 18/06/2019

```
int calcola (int *a, int b, int *c){
  *a = *a * b;          printf("a=%d\n",*a);
  b = b * 2;            printf("b=%d\n",b);
  *c = *a – b;          printf("c=%d\n",*c);
  return *a + b + *c;
}

int main (){
  int x, z;
  int *y;
  y = &x;
  z = -2;
  *y = 3;
  x = 2;
  printf("x=%d y=%d z=%d\n",x,*y,z);
  z = calcola(y,x,&z);
  printf("x=%d y=%d z=%d\n",x,*y,z);
 }
```

x=2 y=2 z=-2

a=4

b=4

c=0

x=4 y=4 z=8

cosa viene stampato dall'esecuzione del programma

1° Appello - Prova scritta – 16/01/2015

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 4
typedef int GGMM[MAX];
typedef int Anno[MAX];
GGMM gm;
Anno a;

int f1 (int anno[ ], int ggmm[ ], int* dim) {
  *dim = anno[(*dim)%MAX];
  printf("dim=%d \n", *dim);
  *dim = (*dim) % MAX;
  printf("dim=%d \n", *dim);
  anno[(*dim)]*=ggmm[(*dim)];
  printf("anno[%d]=%d \n", *dim, anno[(*dim)]);
  return anno[(*dim)]+ggmm[(*dim)];
}
main() {
  int i, k;
  // Inserisci le cifre del tuo anno di nascita, una alla volta
  for (i=0;i< MAX;i++)          {  scanf("%d", &a[i]);  }
  // Inserisci le cifre del tuo mese di nascita, una alla volta
  for (i=2;i< MAX;i++)          {  scanf("%d", &gm[i]);  }
  // Inserisci le cifre del tuo giorno di nascita, una alla volta
  for (i=0;i<2;i++)   {  scanf("%d", &gm[i]);  }
  for (i=0; i< MAX; i++)        {  printf("%d %d ", a[i], gm[i]);   }
  k=gm[1];
  printf("K=%d \n", k);
  printf("Output: %d\n", f1(a, gm, &k));
  printf("K=%d \n", k);
  for (i=0; i< MAX; i++)        {  printf("%d ", a[i]);   }
}
```

[Domanda7] Si scriva esattamente l'output (istruzioni printf), nel riquadro a fianco, che viene prodotto durante l'esecuzione del programma.

1° Appello - Prova scritta – 16/01/2015

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 4
typedef int GGMM[MAX];
typedef int Anno[MAX];
GGMM gm;
Anno a;

int f1 (int anno[ ], int ggmm[ ], int* dim) {
  *dim = anno[(*dim)%MAX];
  printf("dim=%d \n", *dim);
  *dim = (*dim) % MAX;
  printf("dim=%d \n", *dim);
  anno[(*dim)]*=ggmm[(*dim)];
  printf("anno[%d]=%d \n", *dim, anno[(*dim)]);
  return anno[(*dim)]+ggmm[(*dim)];
}
main() {
  int i, k;
  // Inserisci le cifre del tuo anno di nascita, una alla volta
  for (i=0;i< MAX;i++)          {  scanf("%d", &a[i]);  }
  // Inserisci le cifre del tuo mese di nascita, una alla volta
  for (i=2;i< MAX;i++)          {  scanf("%d", &gm[i]);  }
  // Inserisci le cifre del tuo giorno di nascita, una alla volta
  for (i=0;i<2;i++)   {  scanf("%d", &gm[i]);  }
  for (i=0; i< MAX; i++)        {  printf("%d %d ", a[i], gm[i]);   }
  k=gm[1];
  printf("K=%d \n", k);
  printf("Output: %d\n", f1(a, gm, &k));
  printf("K=%d \n", k);
  for (i=0; i< MAX; i++)        {  printf("%d ", a[i]);   }
}
```

[input 19830727]
1 2 9 7 8 0 3 7 K=7
dim=3
dim=3
anno[3]=21
Output: 28
K=3
1 9 8 21