ESERCIZIO: Analisi di un programma

Dato il seguente programma C:

```
#include <stdio.h>
#define D 4
float A(float V[], int k)
{int i;
 float s=0.0;
 for(i=0;i<D; i=i+k)
    {
       V[i]=i;
       s=s+V[i];
    }
 k++;
 return s;
}

main()
{ float V[]={1.5, 2.5, 3.5, 4.5};
 int i, j=2;
 for (i=1; i<D; i=i+1)
    V[i]=V[i-1];
 printf("%f\n", A(V,j));
 for(i=0; i<D; i=i+1)
    printf("%f", V[i]);
printf("%d", j);
}
```

Cosa stampa il programma? La risposta deve essere opportunamente motivata.

Soluzione

Dopo la prima iterazione il vettore V vale

V[D]={1.5,1.5,1.5,1.5}

perché il ciclo copia ciascun elemento nella posizione successiva, quindi il primo elemento viene replicato per tutto il vettore.
La chiamata della funzione **A** restituisce 2 che viene stampato dalla **printf**.
Dopo la chiamata il vettore **V** vale

V[D]={0,1.5,2,1.5}

Il secondo ciclo stampa quindi
0,1.5,2,1.5
Il programma stampa j che non è stata modificata dalla funzione in quanto passata per valore.

2

ESERCIZIO: Analisi di un programma

Indicare che cosa stampa il seguente programma. La risposta deve essere opportunamente motivata.

```
#include <stdio.h>

main()
{     int A[6]={0,0,0,0,0,0};
      int i;

      for (i=0; i<6; i=i+3)
          A[i]= A[i] + 2*i;

      for(i=0; i<6; i++)
          if (A[i])
               printf("%d\n",A[i]);
            else
               printf("%d\n",i-A[i]);
}
```

Soluzione

Dopo il primo ciclo for il vettore A vale:

A=

0	0	0	6	0	0

Il secondo ciclo for stampa quindi:

0	1	2	6	4	5

ESERCIZIO

Scrivere una funzione con la seguente interfaccia **`void Scambia(char s[])`**, che scambi il primo carattere della stringa *s* con l'ultimo.

```
void Scambia(char s[])
{
   int i;
   char c;

   for (i=0; s[i] != '\0'; i++);

   c = s[0];
   s[0] = s[i-1];
   s[i-1] = c;
}
```

ESERCIZIO

Si scriva una funzione ricorsiva che calcoli la somma dei numeri pari fino a n

```
int f(int n);
```

Si scriva inoltre un possibile main.

```
#include <stdio.h>

int f(int n)
{  if (n == 0) return 0;
   else if (n%2 == 1) return f(n-1);
         else return n + f(n-1);
}

void main()
{ int num;
   printf("inserire numero");
   scanf("%d", &num);
   printf("somma pari = %d ", f(num));

   }
```


ESERCIZIO

Si scriva una funzione ricorsiva che calcoli la somma dei primi n numeri pari

```
int f(int n);
```

Si scriva inoltre un possibile main.

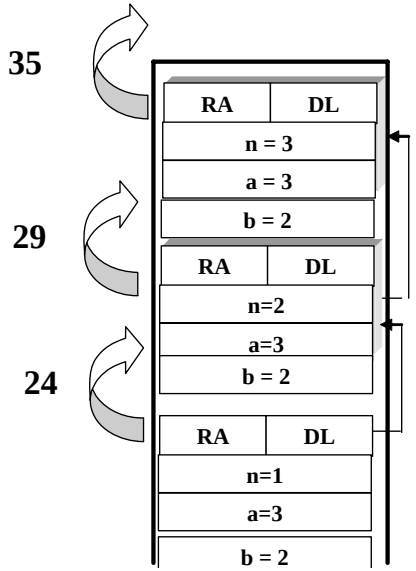
```
#include <stdio.h>

int f(int n)
{  if (n == 0) return 0;
   else return 2*n + f(n-1);
}

void main()
{ int num;
   printf("inserire numero");
   scanf("%d", &num);
   printf("somma pari = %d ", f(num));

   }
```

ESERCIZIO
Data la seguente funzione ricorsiva:

```
double somma_potenza(double a, double b, double n)
{ if (n==1) return a * pow(b,a);
  else return a + n + somma_potenza(a,b,n-1);
}
```

Si dica qual è il valore restituito dalla funzione e si disegnino i record di attivazione nel caso in cui la funzione sia chiamata con i seguenti parametri attuali **`somma_potenza(3,2,3)`**.

Sequenza chiamate
`f(3,2,3)  -> f(3,2,2) -> f(3,2,1)`

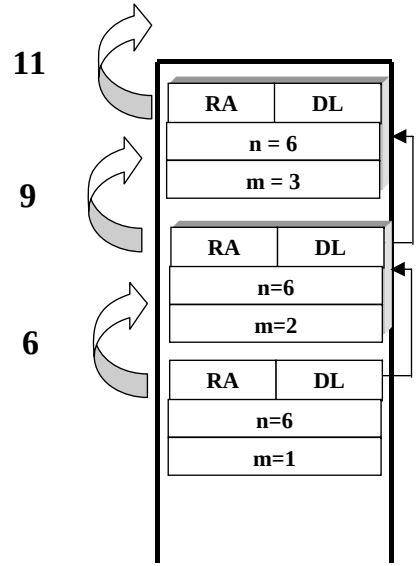


ESERCIZIO
Data la seguente funzione ricorsiva:

```
int f(int m, int n)
{ if (m==1) {return n;}
  else return n/m + f(m-1,n);
}
```

Si dica qual è il valore restituito dalla funzione e si disegnino i record di attivazione nel caso in cui la funzione sia chiamata con i seguenti parametri attuali **`f(3,6)`**.

Sequenza chiamate
`f(3,6)  -> f(2,6) -> f(1,6)`



ESERCIZIO

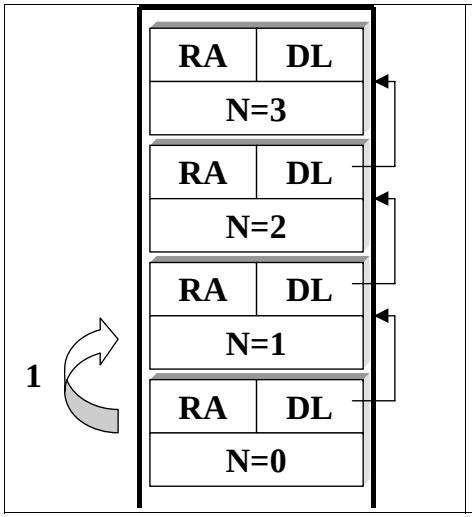
Si consideri la seguente funzione F la cui specifica è data in modo ricorsivo (si supponga N intero):

F(N) = restituisce 1 se N <= 0,
-1+4*F(N-1) + F(N-2), altrimenti

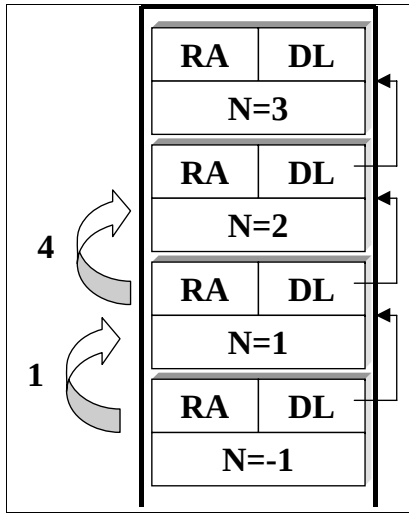
1. Si scriva la funzione C che realizzerebbe tale specifica
2. Si scriva il risultato della funzione quando chiamata con N= 3 e si mostri la sequenza dei record di attivazione;

SOLUZIONE:

```
int F(int N)
{
    if (N<=0) return 1
     else return -1+4*F(N-1) + F(N-2);
}
```

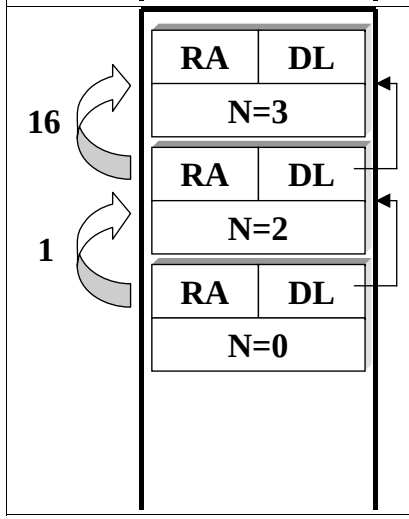


Viene invocata F con N=3, che richiama F(2), che richiama F(1), infine F(0). F(0) restituisce 1 al chiamante



F(1), terminata l'esecuzione di F(0), invoca F(-1), poi termina e fornisce il risultato 4 al chiamante.

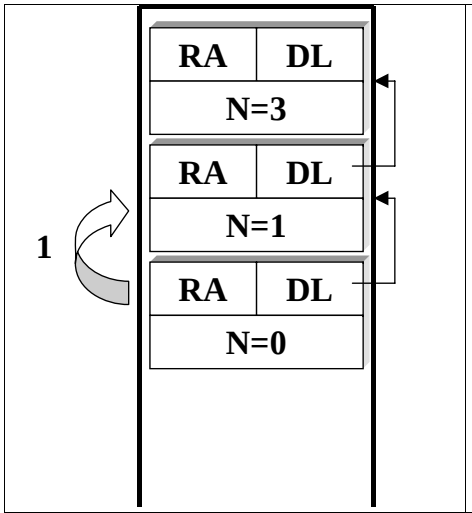
F(1)=4



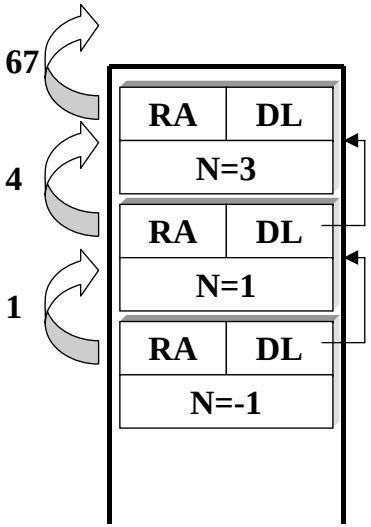
F(2) invoca F(0), che restituisce 1.

F(2)=-1+4*F(1)+F(0)
= -1 + 4*4 +1
=16

F(2) restituisce il valore 16 al chiamante



F(3) invoca F(1), che invoca F(0). Come prima, F(0) restituisce 1.



F(1) invoca F(-1), che restituisce 1. F(1) termina restituendo 4 al chiamante F(3)).

F(3)= -1+4*F(2)+F(1)
=-1+4*16+4
= 67

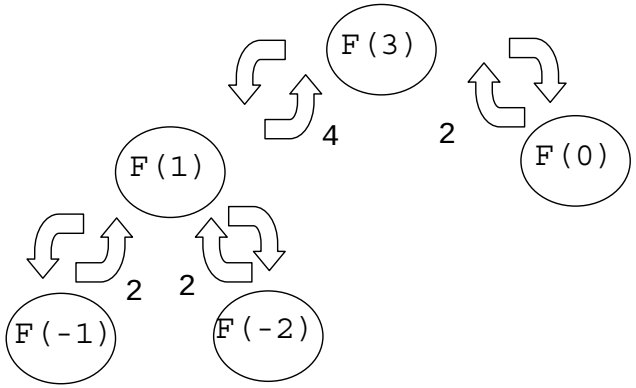
F(3) termina e restituisce 67

ESERCIZIO

Si consideri la seguente funzione F la cui specifica è data in modo ricorsivo (si supponga N intero):
F(N) = restituisce 2 se N <= 0,
F(N-2) * F(N-3), altrimenti
a) Si scriva il risultato della funzione quando chiamata con N= 3 e si mostrino i valori intermedi assunti da N;
b) Si scriva la funzione C che realizzerebbe tale specifica
c) Si mostrino i record di attivazione nello stack

Soluzione:

Sequenza di attivazioni:

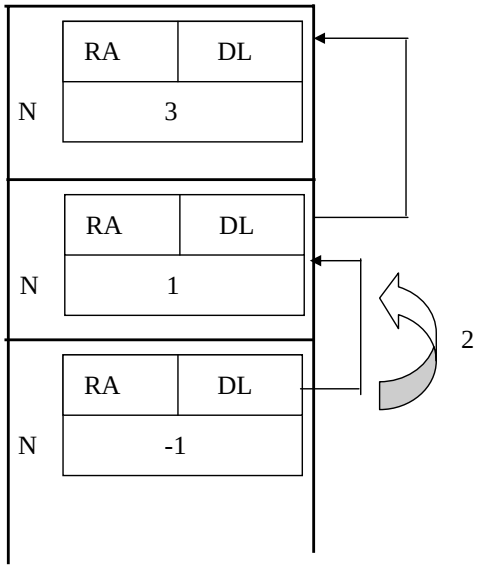


F(3) restituisce il valore **8**

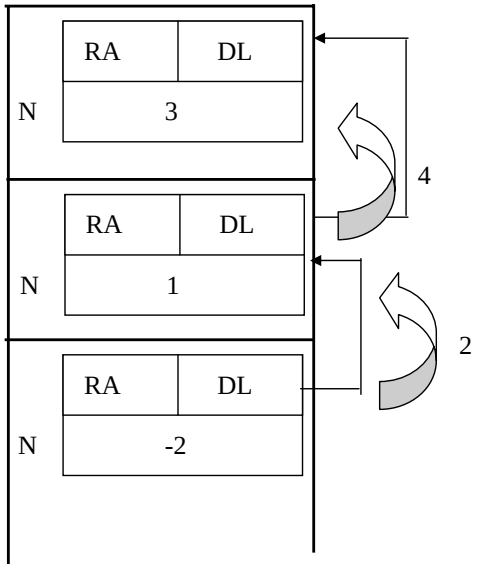
Valori assunti da N: 3 1 -1 -2 0

```
int  F(int N)
{if (N<=0) return 2;
 else  return F(N-2)*F(N-3);
}
```

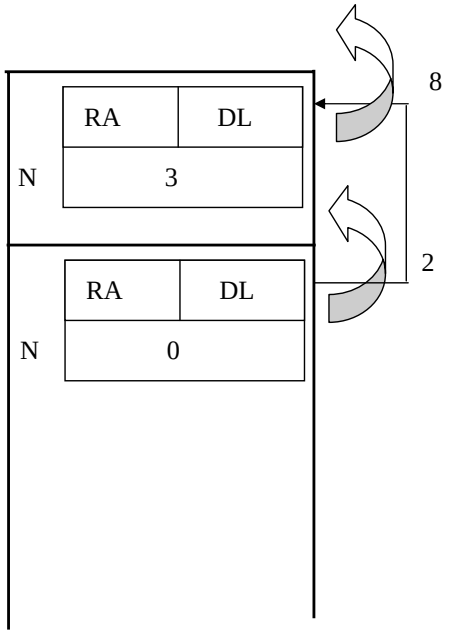
Record di attivazione della chiamata con N = 3



La F(3) chiama la F(1) che a sua volta chiama F(-1). Quest'ultima finisce restituendo il valore 2



Dopo di che la funzione F(1) chiama F(-2). Quest'ultima finisce restituendo il valore 2. Quindi finisce anche F(1) restituendo il valore 4 (risultato della moltiplicazione tra 2 e 2).



Dopo di che la funzione F(3) chiama F(0). Quest'ultima finisce restituendo il valore 2. Quindi finisce anche F(3) restituendo il valore 8 (risultato della moltiplicazione tra 4 e 2).

ESERCIZIO

Si consideri la seguente procedura P la cui specifica è data in modo ricorsivo (si supponga N intero):

P(*N*) = stampa *N* se *N* <= 10,
stampa *N* ed invoca P(*N*-10), altrimenti

a) Scrivere il codice C di tale procedura.

b) Si scriva la sequenza di valori stampati quando la procedura è chiamata con *N*= 30. Si mostri anche la sequenza dei record di attivazione.

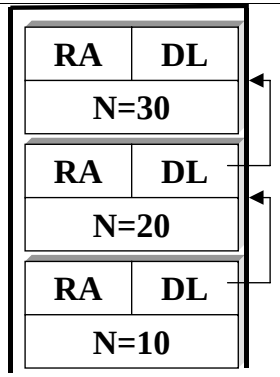
Soluzione:

a) Codice della procedura

```
void P(int N)
{ printf("%d ",N);
  if (N>10)
    { P(N-10); }
}
```

b) La procedura stampa la sequenza

30 20 10



La procedura stampa 30, poi invoca P(20), che stampa il valore 20 ed invoca P(10). P(10) stampa 10 e termina. Il controllo torna a P(20) che termina, poi a P(30) che termina.

ESERCIZIO

Scrivere una procedura in C che lavora su due parametri uno di ingresso e uno di uscita.

```
void prod(int a, int *b);
```

La procedura deve produrre nel secondo parametro **b** il valore di **a*a** calcolato come sequenza di somme

```
void prod(int a, int *b){
   int i, P=0;
   for(i=1; i<=a; i++)
      P = P + a;
   *b = P;
}
```

Un possibile main

```
void main()
{
   int num, quad;
   printf("inserisci un numero");
   scanf("%d", &num);
   prod(num, &quad);
   printf("quadrato  =  %d", quad);
}
```


ESERCIZIO

Scrivere una procedura in C che lavora su quattro parametri due di ingresso e due di uscita. I due parametri di ingresso rappresentano un vettore e la sua dimensione, mentre i parametri di uscita il valore minimo e massimo del vettore.

```
void minmax(int V[], int dim, int *min, int *max);
```

```
void minmax(int V[], int dim, int *min, int *max)
{ int i, minimo=V[0], massimo= V[0];

   for (i=1; i<dim; i++)
      {if(V[i] < minimo) minimo = V[i];
       if(V[i] > massimo) massimo = V[i];
      }
   *min = minimo;
   *max = massimo;
}
```

Un possibile main

```
void main()
{
   int vett[5], min, max, i;
   for(i=0; i<5; i++)
{  printf("inserisci un numero");
      scanf("%d", &vett[i]);
}
   minmax(vett, 5, &min, &max);
   printf("minimo = %d\n massimo = %d", min,max);
}
```

ESERCIZIO

Data la struttura

```
struct grandezza{char nome[30];
            char unita_misura[10];
            int valore;}
```

scrivere una procedura che riceve in ingresso un array G di strutture **`struct grandezza`**, la sua dimensione **`Dim`** e un valore di soglia **`soglia`** e stampa restituisce in **`max_dist`** il massimo tra delle distanze tra ogni valore **`valore`** contenuto nel'array e la **`soglia`**.

```
 void dist_max(struct grandezza G[], int Dim, int
soglia, int* max_dist);
```

Quale sarebbe l'interfaccia di una funzione che effettua il medesimo calcolo e restituisce la massima distanza ?

```
void dist_max(struct grandezza G[], int Dim, int
soglia, int* max_dist)
{int i, max = G[0].valore - soglia;
 for (i=1; i < Dim; i++)
    {if (max < G[i].valore - soglia)
        max = G[i].valore – soglia;
    }
*max_dist = max;
}
```

Interfaccia funzione

```
int dist_max(struct grandezza G[], int Dim, int soglia)
```


Esercizio

Si scriva una funzione ricorsiva

`int r(int a, int b);` che calcoli il seguente valore

$$\sum_{i=1}^{b} (a^i - i)$$

Si supponga di avere a disposizione una funzione

`int potenza(int x, int y);`

che calcoli x^y

Soluzione

```
int r(int a, int b)
{ if (b == 1)
      return a-1;
  else return potenza(a,b)-b+r(a,b-1);
}
```

Esercizio

Si scriva una funzione ricorsiva
`int fun(int d, int s, int o);` che calcoli il seguente valore

$$\prod_{i=0}^{d} (s + o * i)$$

```
int fun(int d, int s, int o)
{if (d==0) return s;
   else return (s + o*d)*fun(d-1, s, o);
}
```