Fondamenti di Informatica L-A (A.A. 2002/2003) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista - Appello del 13/11/2002 - durata 1h45m - COMPITO A

- **Esercizio 1 (punti 7)**

Dato il seguente programma C:

```
#include <stdio.h>

int f1(int b, int y) {
      if (b%2) return b;
      else { b++; return y; } }
main() {
  int a=-8, b=0;
  while (a++ < 0) { a+=2; b = f1(a,b);
      printf("%d\n", b++); }
}
```

Che cosa fa la funzione f1? Che cosa stampa il programma (motivare opportunamente la risposta)?

- **Esercizio 2**

- Si definisca una funzione **`int sum (int i)`** (in maniera ricorsiva o iterativa) che, passato un valore intero **`i`**, calcoli la sommatoria dei numeri interi da **`1`** a **i**: **(punti 3)**

$$\sum_{k=1}^{i} k$$

- Si scriva poi una funzione **ricorsiva `int prodsum(int k)`** che calcoli il seguente valore:

$$\prod_{i=1}^{m} \sum_{k=1}^{i} k$$

chiamando opportunamente al suo interno la funzione **`sum()`**, precedentemente definita **(punti 3)**.

- Si scriva poi il programma **`main`** e si stampi il risultato per un valore **`N`** introdotto a terminale dall'utente **(punti 3)**.

- **Esercizio 3 (punti 8)**

Si consideri la seguente funzione F:

```
int F(double x){
      if (x<2) return 1;
      else return F(x-2);
      x++;}
```

Si scriva il risultato della funzione quando invocata come F(7) e si mostrino i record di attivazione. La funzione è tail-ricorsiva?

- **Esercizio 4 (punti 3)**

Si consideri la grammatica **G** con scopo **`S`** e simboli terminali **`{a,z,0,1}`**

```
S::= C B C | C
C::= B A | A
B::= a | z
A::= 0 | 1
```

Si dica se la stringa **`a1z0`** è sintatticamente corretta rispetto a tale grammatica e se ne mostri la derivazione *left most*.

- **Esercizio 5 (punti 2)**

Dato il seguente programma quale sarà il risultato stampato? Si motivi bene la risposta data.

```
int piu(int *n) {return *n=*n + *n;}
main () { int n=12;
printf("%d", piu(&n));
printf("%d", n); }
```

- **Esercizio 6 (punti 3)**

Un elaboratore rappresenta i numeri interi su 8 bit dei quali 7 sono dedicati alla rappresentazione del modulo del numero e uno al suo segno. Indicare come viene svolta la seguente operazione aritmetica e determinarne il risultato traslandolo poi in decimale per la verifica: **62+35**

Fondamenti di Informatica L-A (A.A. 2002/2003) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista - Appello del 13/11/2002 - durata 1h45m - COMPITO B

- **Esercizio 1 (punti 7)**

Dato il seguente programma C:

```
#include <stdio.h>

int g1(int b, int y) {
      if (b%3) return b;
      else { b++; return y; } }
main() {
  int a=-8, b=0;
  for (a=-8; a++ < 0; a+=1) { b = g1(a,b);
      printf("%d\n", b++); }
}
```

Che cosa fa la funzione g1? Cosa stampa il programma (motivare opportunamente la risposta)?

- **Esercizio 2**

- Si definisca una funzione **`int prod (int i)`** (in maniera ricorsiva o iterativa) che passato un valore intero i, calcoli la produttoria dei numeri interi da **`1`** a **`i`** **(punti 3):**

$$\prod_{k=1}^{i} k$$

- Si scriva poi una funzione **ricorsiva `int sumprod (int k)`** che calcoli il seguente valore:

$$\sum_{i=1}^{m} \prod_{k=1}^{i} k$$

chiamando opportunamente al suo interno la funzione **`prod`**, precedentemente definita **(punti 3).**

- Si scriva poi il programma **`main`** e si stampi il risultato per un valore **`N`** introdotto a terminale dall'utente **(punti 3).**

- **Esercizio 3 (punti 8)**

Si consideri la seguente funzione G:

```
int G(double x){
      if (x<-1) return 1;
      else {x-=2; return G(x)/2;}
      }
```

Si scriva il risultato della funzione quando invocata come G(4) e si mostrino i record di attivazione. La funzione è tail-ricorsiva?

- **Esercizio 4 (punti 3)**

Si consideri la grammatica G con scopo S e simboli terminali **`{1,0,x,y}`**

```
S::= E | E B E
E::= A B | A
B::= 0 | 1
A::= x | y
```

Si dica se la stringa **`x01y`** è sintatticamente corretta rispetto a tale grammatica e se ne mostri la derivazione *left most*.

- **Esercizio 5 (punti 2)**

Dato il seguente programma quale sarà il risultato stampato? Si motivi bene la risposta data.

```
int per(int *n) {return *n=*n * *n;}
main () { int n=5;
printf("%d", per(&n));
printf("%d", n); }
```

- **Esercizio 6 (punti 3)**

Un elaboratore rappresenta i numeri interi su 8 bit dei quali 7 sono dedicati alla rappresentazione del modulo del numero e uno al suo segno. Indicare come viene svolta la seguente operazione aritmetica e determinarne il risultato traslandolo poi in decimale per la verifica:

Soluzione Compito A

Esercizio 1

Il programma stampa a video la sequenza di interi, ciascuno su una linea diversa:
-5 -4 1

Infatti, la prima esecuzione del ciclo avviene con a inizialmente uguale a -7 e poi immediatamente incrementato di 2 (prima istruzione del ciclo). La funzione f1() viene quindi invocata con parametri attuali -5, 0. f1() non fa altro che restituire il primo parametro in caso di primo parametro dispari, il secondo parametro altrimenti (l'incremento di b prima di return y non può influire in alcun modo sul parametro b del chiamante). printf() stampa il valore di b nel main() e quindi -5, e l'incremento di b avviene solo dopo la stampa.
La seconda esecuzione del ciclo comincia dunque con a=-4 e b=-4; a viene subito incrementato di 2 e f1() viene invocata con parametri attuali -2 e -4. Poiché il primo parametro di invocazione è pari, f1() restituisce -4; tale valore viene assegnato a b nel main() e stampato a video.
La terza esecuzione del ciclo comincia dunque con a=-1 e b=-4; a viene subito incrementato di 2 e f1() viene invocata con parametri attuali 1 e -4. Poiché il primo parametro di invocazione è dispari, questa volta f1() restituisce 1; tale valore viene assegnato a b nel main() e stampato a video.
La variabile a ha ora valore +1, la condizione del ciclo while non è più verificata, quindi il corpo del ciclo non è eseguito e il programma termina.

Esercizio 2

```
double sum(int i)
{ if (i==0) return 0;
  else return i + sum(i-1);
}
```

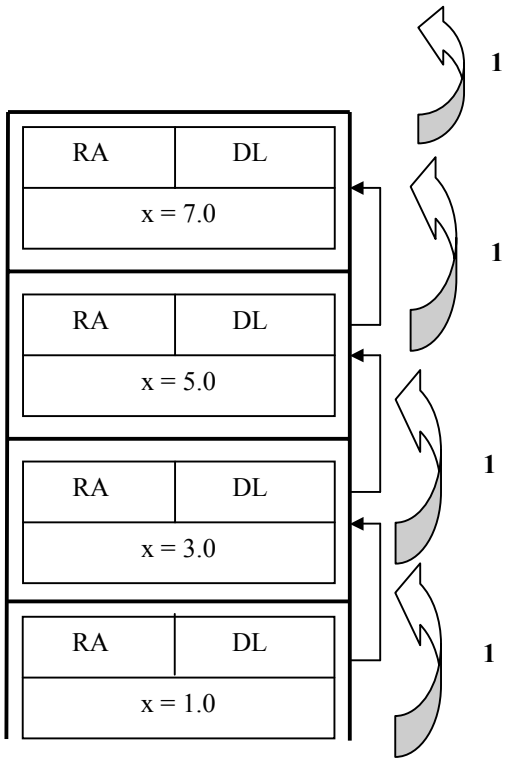
```
double prodsum(int k)
{ if (k==1) return sum(1);
  else return sum(k)* prodsum(k-1);}
```

```
#include <stdio.h>

main()
{
      int n;
      printf("Inserisci il numero ");
      scanf("%d", &n);
    printf("Risultato: %d\n",prodsum(n));

}
```

Esercizio 3
Viene stampato a video 1.



La funzione è tail ricorsiva.

Esercizio 4
La stringa è sintatticamente corretta.
Derivazione *left most*:
S → C B C → B A B C → B A B A → a A B A → a 1 B A → a 1 z A → a 1 z 0

Esercizio 5
24 24
Tutto avviene come se il passaggio fosse per indirizzo, la variabile n del programma chiamante viene modificata.

Esercizio 6
62-> 0 0111110
35-> 0 0100011
Tra i moduli dei numeri si esegue una somma:

```
  0111110
+ 0100011
  ---------
  1100001
```

che vale 97 in base dieci.

Soluzione Compito B

Esercizio 1

Il programma stampa a video la sequenza di interi, ciascuno su una linea diversa:
-7 -5 -4 -1

Infatti, la prima esecuzione del ciclo avviene con a uguale a -8 (durante la valutazione della condizione) e poi immediatamente incrementato di 1 (post-incremento nella condizione di prosecuzione del ciclo for). La funzione g1() viene quindi invocata con parametri attuali -7, 0. g1() non fa altro che restituire il primo parametro in caso di primo parametro non divisibile per 3, il secondo parametro altrimenti (l'incremento di b prima di return y non può influire in alcun modo sul parametro b del chiamante). printf() stampa il valore di b nel main() e quindi -7, e l'incremento di b avviene solo dopo la stampa.
La seconda esecuzione del ciclo comincia dunque con a=-6 e b=-6; a viene subito incrementato (post-incremento nella condizione di prosecuzione del ciclo for) e g1() viene invocata con parametri attuali -5 e -6. Poiché il primo parametro di invocazione è ancora non divisibile per 3, g1() restituisce -5; tale valore viene assegnato a b nel main() e stampato a video.
La terza esecuzione del ciclo comincia dunque con a=-4 e b=-5; a viene subito incrementato e g1() invocata con parametri attuali -3 e -4. Poiché il primo parametro di invocazione è questa volta divisibile per 3, f1() restituisce -4; tale valore viene assegnato a b nel main() e stampato a video.
La quarta e ultima esecuzione del ciclo comincia dunque con a=-2 e b=-4; a viene subito incrementato e g1() invocata con parametri attuali -1 e -4. Poiché il primo parametro di invocazione non è divisibile per 3, f1() restituisce -1; tale valore viene assegnato a b nel main() e stampato a video. La variabile a ha ora valore 0, la condizione del ciclo while non è più verificata, quindi il corpo del ciclo non è eseguito e il programma termina.

Esercizio 2

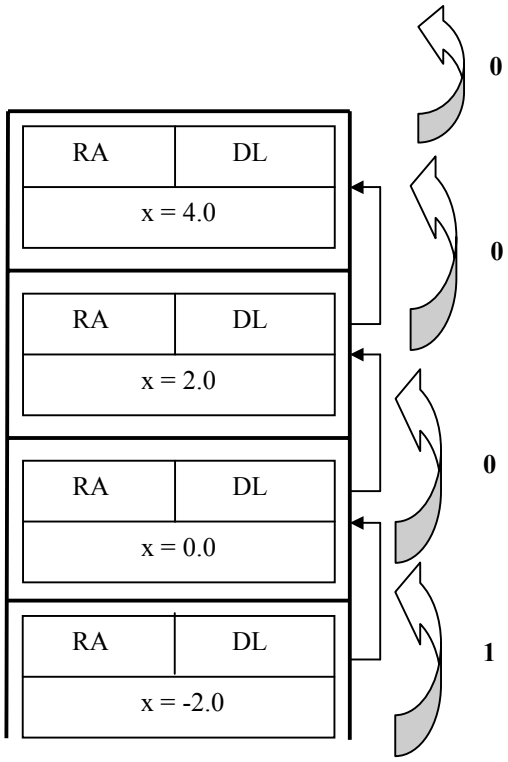
```
double prod(int k)
{ if (k==1) return 1;
  else return k * prod(k-1);
}
```

```
double sumprod (int k)
{ if (k==1) return prod(1);
  else return prod(k) + sumprod(k-1);}
```

```
#include <stdio.h>

main()
{
      int n;
      printf("Inserisci il numero ");
      scanf("%d", &n);
     printf("Risultato: %d\n",sumprod(n));
}
```

Esercizio 3
Viene stampato a video 0.



La funzione non è tail-ricorsiva perché la chiamata ricorsiva non è l'ultima operazione svolta dalla funzione G(). Infatti, dopo il ritorno del controllo dalla funzione chiamata, la funzione G() deve ancora effettuare la divisione intera del risultato ricevuto per 2, prima di restituire il quoziente intero al chiamante.

Esercizio 4
La stringa è sintatticamente corretta.
Derivazione *left most*:
$S \rightarrow E\,B\,E \rightarrow A\,B\,B\,E \rightarrow A\,B\,B\,A \rightarrow x\,B\,B\,A \rightarrow x\,0\,B\,A \rightarrow x\,0\,1\,A \rightarrow x\,0\,1\,y$

Esercizio 5
25 25
Tutto avviene come se il passaggio fosse per indirizzo, la variabile n del programma chiamante viene modificata.

Esercizio 6
118 -> 0 1110110
-37-> 1 0100101
Tra i moduli dei numeri si esegue una sottrazione:

```
    1110110
-   0100101
    ---------
    1010001
```

che vale 81 in base dieci.