Fondamenti di Informatica L-A (A.A. 2003/2004) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – I Prova Intermedia del 05/11/2003 - durata 2h - COMPITO B

- **Esercizio 1 (punti 4)**

Dato il seguente programma C:

```
#include <stdio.h>

int mul(int x, int y) {
      if ((x%5) != 0 )
            return mul(x-1, y);
      else
            return ++y;
}

main() {
      int a = 15;
      int b = 1;

      while (a > 0)
      {
            b = mul(a, b);
            printf("%d\n", b);
            a-=5;
      }
}
```

Che cosa fa la funzione mul()? Che cosa stampa il programma? Si motivi opportunamente la risposta.

- **Esercizio 2 (punti 16)**

a) Si definisca una funzione ricorsiva **`long fattoriale (long n)`** che calcoli in maniera ricorsiva il fattoriale del parametro n. Si definisca inoltre una funzione iterativa **`float potenza (float num, long exp)`** che calcoli in maniera iterativa la potenza del parametro num elevato ad exp. **(punti 4)**

b) Si definisca una funzione **`float F(float x, float y, long n)`** che calcoli il seguente valore: **(punti 6)**

$$\prod_{k=0}^{n} g(n,k)x^{n-k}y^{k}$$

dove la funzione g(n, k) è così definita:

$$g(n,k)=\frac{k!(n+k)!}{n!}$$

c) Si scriva infine un programma main che chieda all'utente di inserire tre valori, x e y (float), e n (int). Il programma controlli, in particolare, che n sia strettamente maggiore di zero (in caso contrario si continui a chiedere all'utente un nuovo valore finché questi non soddisfa la condizione posta). Il programma stampi infine il valore ottenuto invocando la funzione F() con parametri x, y, n e poi termini. **(punti 6).**

- **Esercizio 3 (punti 5)**

Si consideri la seguente funzione dfg():

```
int dfg(int x, int y) {
      if ((x/y) >= 3)
            return 1 + dfg(x-3, y);
      else
            return (y-1);
}
```

Si scriva il risultato della funzione quando invocata come dfg(16, 3) e si mostrino i record di attivazione. La funzione è tail-ricorsiva?

- **Esercizio 4 (punti 2)**

Si consideri la grammatica **G** con scopo **S** e simboli terminali **{to,po,li,no}**

```
S::= A | A B
A::= to | po | li | no
B::= C A D | A D C
C::= po
D::= no
```

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

- **Esercizio 5 (punti 3)**

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

```
int sum(int * x, int y) {
      *x = *x  + y;
      return *x; }

main () { int a=3, b=4;
printf("%d\n", sum(&a,b));
printf("%d\n", a); }
```

- **Esercizio 6 (punti 2)**

La seguente funzione vorrebbe calcolare la somma di tutti i numeri che vanno da 0 ad a. Che cosa c'è di sbagliato nel codice? Se ne proponga anche una versione corretta.

```
int sommatoria(int a) {
  return a + sommatoria(a-1); }
```

Soluzione Compito B

Esercizio 1

Il programma stampa a video la sequenza di interi, ciascuno su una linea diversa:
2 3 4

Il main contiene come prime due istruzioni le dichiarazioni delle variabili a e b, inizializzate rispettivamente ai valori 15 e 1. Subito dopo si trova un ciclo while, il cui test di condizione è verificato; quindi vengono eseguite le istruzioni all'interno del ciclo.
Alla variabile b viene assegnato il valore di ritorno della funzione mul(), invocata con parametri a e b, che valgono rispettivamente 15 e 1 (sono ancora i valori iniziali). La funzione mul() restituisce il risultato della chiamata ricorsiva con parametri x-1, y se l'attuale valore di x non è un multiplo esatto di 5, altrimenti restituisce l'attuale valore di y incrementato di 1. Siccome alla prima invocazione la variabile x vale 15, la condizione di test non è verificata e la funzione restituisce immediatamente il valore di y incrementato (da notare che si tratta di un pre-incremento). Siccome y vale 1, la funzione restituisce il valore 2, che viene assegnato alla variabile b nel main. A questo punto la printf() provvede a stampare il valore di b, che per quanto appena detto è 2. Poi la variabile a viene decrementata di 5 (il suo nuovo valore è 10), e si ricomincia il ciclo.

Alla seconda iterazione la condizione di ingresso nel ciclo è ancora verificata, e le variabili a e b valgono rispettivamente 10 e 2. Viene ripetuta l'invocazione alla funzione mul(), la quale ritorna il valore y pre-incrementato di 1 (e quindi questa volta vale 3). Il valore 3 viene assegnato alla variabile b, e viene stampato a video dalla printf(). Infine il valore della variabile a viene decrementato di 5. Al termine della seconda iterazione quindi a vale 5 e b vale 3.

Alla terza iterazione la condizione di ingresso nel ciclo è ancora soddisfatta, e alla variabile b viene di nuovo assegnato il valore di ritorno della funzione mul(): siccome mul() viene invocata con parametri attuali x=5 e y=3, e siccome x è un multiplo esatto di 5, mul() restituisce il valore di y pre-incrementato, cioè restituisce 4. Tale valore viene assegnato nel main alla variabile b, che poi viene subito stampata dalla printf(). Infine il valore di a è decrementato di 5 e va a 0.

A questo punto la condizione di ingresso nel ciclo non è più soddisfatta, e quindi il programma termina.

Esercizio 2

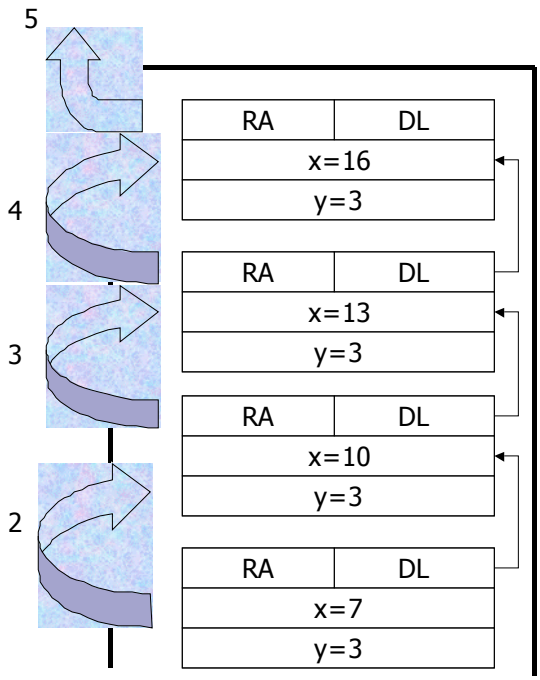
```
long fattoriale(long n) {
     if (n == 0)
          return 1;
     else
          return n * fattoriale(n-1);
}

float potenza(float num, long exp) {
     float result = 1;
     for ( ; exp>0; exp--)
          result = result * num;
     return result;
}
```

```
long g(long n, long k) {
     return (fattoriale(k) * fattoriale(n+k)) / fattoriale(n);
}

float F(float x, float y, long n) {
     long i;
     float result = 1;
     for (i=0; i<=n; i++) {
          result = result * g(n, i)* potenza(x, n-i) * potenza(y, i);
     }
     return result;
}
```

```
main() {
     float x;
     float y;
     long n;

     printf("Inserire x, y e n: ");
     scanf("%f %f %d", &x, &y, &n);
     while (n<=0) {
          printf("Re-inserire n: ");
          scanf("%d", &n);
     }
     printf("F(%f, %f, %d) = %f\n", x, y, n, F(x, y, n));
}
```

Esercizio 3

Il risultato è 5.



La funzione NON è tail ricorsiva.

Esercizio 4

La stringa è sintatticamente corretta.
Derivazione *left most*:
S → A B → to B → to C A D → to po A D → to po li D → to po li no

Esercizio 5
7 7
La funzione riceve il parametro x per riferimento ed il parametro y per copia, e produce side-effects sulla variabile a. In particolare, a x viene assegnato il valore della somma di y con il valore iniziale di x. Questo assegnamento modifica anche il valore di a, per quanto detto sopra. La funzione ritorna il valore di x, che vale appunto 7. La seconda istruzione printf() stampa a video il valore attuale di a che, a causa del side-effect nella funzione sum(), vale 7.

Esercizio 6

La funzione è definita ricorsivamente, ma manca la condizione di terminazione della ricorsione: così com'è questa funzione non termina mai. Una versione corretta potrebbe essere:

```
int sommatoria(int a) {
  if (a == 0)
      return 0;
  else
      return a + sommatoria(a-1); }
```