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 A

- **Esercizio 1 (punti 4)**

Dato il seguente programma C:

```
#include <stdio.h>

int fun(int a, int b) {
      if ((a%3) == 0 )
            return ++b;
      else
            return fun(a-1, b);
}

main() {
      int x = 9;
      int y = 0;

      while (x > 0)
      {
            y = fun(x, y);
            printf("%d\n", y);
            x-=3;
      }
}
```

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

- **Esercizio 2 (punti 16)**

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

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

$$\sum_{k=0}^{n} \binom{n}{k} a^{n-k} b^{k}$$

dove il simbolo del binomio è così definito:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

c) Si scriva infine un programma main che chieda all'utente di inserire tre valori, a e b (float), e n (int). Il programma controlli, in particolare, che n sia maggiore o uguale a 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 newton() con parametri a, b, n e poi termini. **(punti 6).**

- **Esercizio 3 (punti 5)**

Si consideri la seguente funzione g():

```
int g(int a, int b) {
      if ((a/b) <2 )
            return (b-1);
      else
            return 1 + g(a-2, b);
}
```

Si scriva il risultato della funzione quando invocata come g(9, 2) 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 **{pa,pe,ri,no}**

```
S::= B C | A B C
A::= pa
B::= pa | pe | ri | no
C::= D B | B
D::= ri
```

Si dica se la stringa **paperino** 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 prod(int * a, int * b) {
      *a = *a  *  *b;
      return *a; }

main () { int x=3, y=4;
printf("%d\n", prod(&x,&y));
printf("%d\n", x); }
```

- **Esercizio 6 (punti 2)**

La seguente funzione vorrebbe calcolare il valore di 2 elevato a potenza con esponente pari ad a. Che cosa c'è di sbagliato nel codice? Se ne proponga anche una versione corretta.

```
int funz(int a) {
  return 2 * funz(a-1); }
```

Soluzione Compito A

Esercizio 1

Il programma stampa a video la sequenza di interi, ciascuno su una linea diversa:
1 2 3

Il main contiene come prime due istruzioni le dichiarazioni delle variabili x e y, inizializzate rispettivamente ai valori 9 e 0. 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 y viene assegnato il valore di ritorno della funzione fun(), invocata con parametri x e y che valgono rispettivamente 9 e 0 (sono ancora i valori iniziali). La funzione fun() restituisce il valore di b incrementato di 1 se il valore di a è un multiplo esatto di 3, altrimenti invoca ricorsivamente fun() con a decrementato di 1. Siccome alla prima invocazione la variabile a vale 9, la condizione di test è verificata e la funzione restituisce immediatamente il valore di b incrementato (da notare che si tratta di un pre-incremento). Siccome b vale 0, la funzione restituisce il valore 1, che viene assegnato alla variabile y nel main. A questo punto la printf() provvede a stampare il valore di y, che per quanto appena detto è 1. Poi la variabile x viene decrementata di 3 (il suo nuovo valore è 6), e si ricomincia il ciclo.

Alla seconda iterazione la condizione di ingresso nel ciclo è ancora verificata, e le variabili x e y valgono rispettivamente 6 e 1. Viene ripetuta l'invocazione alla funzione fun(), la quale ritorna il valore b pre-incrementato di 1 (e quindi questa volta vale 2). Il valore 2 viene assegnato ad y, e viene stampato a video dalla printf(). Infine il valore della variabile x viene decrementato di 3. Al termine della seconda iterazione quindi x vale 3 e y vale 2.

Alla terza iterazione la condizione di ingresso nel ciclo è ancora soddisfatta, e a y viene di nuovo assegnato il valore di ritorno della funzione fun(): siccome fun() viene invocata con parametri attuali a=3 e b=2, e siccome a è un multiplo esatto di 3, fun() restituisce il valore di b pre-incrementato, cioè restituisce 3. Tale valore viene assegnato nel main alla variabile y, che poi viene subito stampata dalla printf(). Infine il valore di x è decrementato di 3 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 fact(long n) {
     long result = 1;
     for ( ; n>0; n--)
          result = result * n;
     return result;
}
```

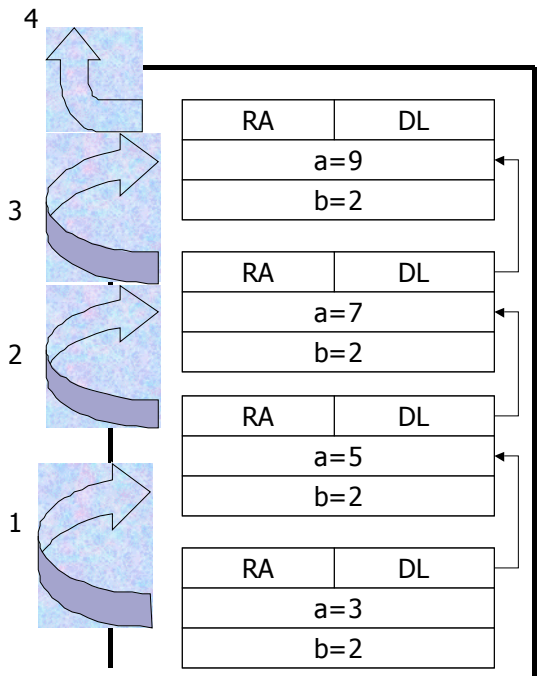
```
float power(float base, long esp) {
     if (esp == 0)
          return 1;
     else
          return base * power(base, esp-1);
}
```

```
long binomio(long n, long k) {
     return fact(n)/(fact(k) * fact(n-k));
}

float newton(float a, float b, long n) {
     long i;
     float result = 0;
     for (i=0; i<=n; i++)
          result = result + binomio(n, i) * power(a, n-i) * power(b, i);
     return result;
}
```

```
main() {
     float a;
     float b;
     long n=-1;

     printf("Inserire a, b e n: ");
     scanf("%f %f %d", &a, &b, &n);
     while (n<0) {
          printf("Re-inserire n: ");
          scanf("%d", &n);
     }
     printf("(%f + %f)^%d = %f\n", a, b, n, newton(a, b, n));
}
```

Esercizio 3

Il risultato è 4.



La funzione NON è tail ricorsiva.

Esercizio 4

La stringa è sintatticamente corretta.
Derivazione *left most*:
S → A B C → pa B C → pa pe C → pa pe D B → pa pe ri B → pa pe ri no

Esercizio 5
12 12
La funzione riceve i parametri per riferimento, ed produce side-effects sulla variabile a. In particolare ad essa viene assegnato il valore del prodotto di b per il valore iniziale di a. Questo assegnamento modifica anche il valore di x, per quanto detto sopra. La funzione ritorna il valore di a, che vale appunto 12. La seconda istruzione printf() stampa a video il valore attuale di x che, a causa del side-effect nella funzione prod(), vale 12.

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 funz(int a) {
  if (a == 1)
      return 2;
  else
      return 2 * funz(a-1); }
```