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 C

- **Esercizio 1 (punti 4)**

Dato il seguente programma C:

```
#include <stdio.h>

int calc(int e, int f) {
      if ((e%4) == 0 )
            return ++f;
      else
            return calc(e-1, f);
}

main() {
      int a = 12;
      int b = 2;

      while (a > 0)
      {
            b = calc(a, b);
            printf("%d\n", b);
            a-=4;
      }
}
```

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

- **Esercizio 2 (punti 16)**

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

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

$$\sum_{k=0}^{n} \binom{n}{k} \left(c^{n-k} + d^{k} \right)$$

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, `c` e `d` (float), ed `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 G() con parametri `c, d, n`, e poi termini. **(punti 6).**

- **Esercizio 3 (punti 5)**

Si consideri la seguente funzione abc():

```
int abc(int e, int f) {
      if ((f/e) <= 2 )
            return (e-1);
      else
            return 1 + abc(e, f-4);
}
```

Si scriva il risultato della funzione quando invocata come abc(4, 21) 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,ro,ne}**

```
S::= Y Z | X Y Z
Y::= pa | pe | ro | ne
Z::= W Y | Y
X::= pa
W::= ro
```

Si dica se la stringa **paperone** 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 diff(int * a, int * b) {
      *b = *b  -  *a;
      return *b; }

main () { int x=3, y=6;
printf("%d\n", diff(&x,&y));
printf("%d\n", y); }
```

- **Esercizio 6 (punti 2)**

Si scriva qual è il risultato della invocazione della seguente funzione con parametri attuali x=12.00 e y=5.00. Si motivi opportunamente la risposta fornita.

```
int op(float x, float y) {
  return x/y; }
```

Soluzione Compito C

Esercizio 1

Il programma stampa a video la sequenza di interi, ciascuno su una linea diversa:
3 4 5

Il main contiene come prime due istruzioni le dichiarazioni delle variabili a e b, inizializzate rispettivamente ai valori 12 e 2. 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 calc(), invocata con parametri a e b che valgono rispettivamente 12 e 2 (sono ancora i valori iniziali). La funzione calc() restituisce il valore di f incrementato di 1 se il valore di e è un multiplo esatto di 4, altrimenti invoca ricorsivamente calc() con e decrementato di 1. Siccome alla prima invocazione la variabile e vale 12, la condizione di test è verificata e la funzione restituisce immediatamente il valore di f incrementato (da notare che si tratta di un pre-incremento). Siccome f vale 2, la funzione restituisce il valore 3, 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 3. Poi la variabile a viene decrementata di 4 (il suo nuovo valore è 8), 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 8 e 3. Viene ripetuta l'invocazione alla funzione calc(), la quale ritorna il valore f pre-incrementato di 1 (e quindi questa volta vale 4). Il valore 4 viene assegnato alla variabile b, e viene stampato a video dalla printf(). Infine il valore della variabile a viene decrementato di 4. Al termine della seconda iterazione quindi a vale 4 e b vale 4.

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

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

Esercizio 2

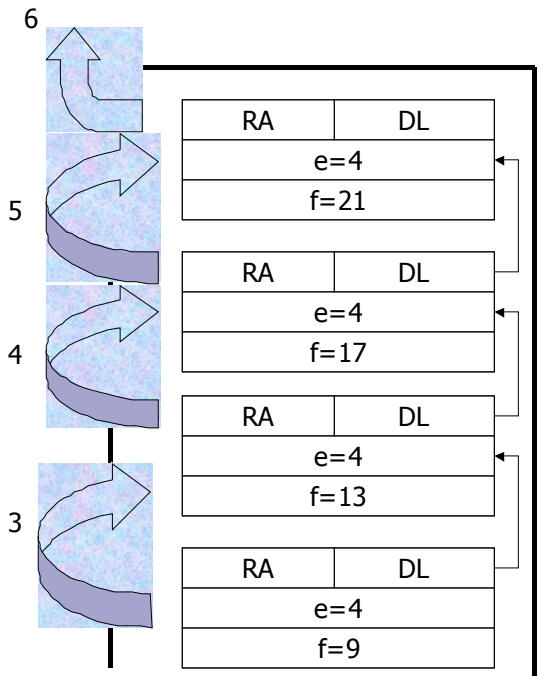
```
long f_fat(long num) {
     long result = 1;
     for ( ; num>0; num--)
          result = result * num;
     return result;
}

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

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


float G(float c, float d, long n) {
     long i;
     float result = 0;
     for (i=0; i<=n; i++) {
          result = result + binomio(n, i) *(f_pow(c, n-i)+f_pow(d, i));
     }
     return result;
}
```

```
main() {
     float c;
     float d;
     long n;

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

Esercizio 3

Il risultato è 6.



La funzione NON è tail ricorsiva.

Esercizio 4

La stringa è sintatticamente corretta.
Derivazione *left most*:
S → X Y Z → pa Y Z → pa pe Z → pa pe W Y → pa pe ro Y → pa pe ro ne

Esercizio 5
3 3
La funzione riceve i parametri per riferimento, ed produce side-effects sulla variabile b. In particolare ad essa viene assegnato il valore della differenza di b meno il valore iniziale di a. Questo assegnamento modifica anche il valore di y, per quanto detto sopra. La funzione ritorna il valore di b, che vale appunto 3. La seconda istruzione printf() stampa a video il valore attuale di y che, a causa del side-effect nella funzione diff(), vale 3.

Esercizio 6

La funzione calcola la divisione tra i due parametri attuali e, siccome la dichiarazione prevede che il suo tipo di ritorno sia un int, il risultato della divisione (float) viene automaticamente trasformato in un intero, troncando la parte frazionaria del numero. Nel caso di x=12.00 e y=5.00, la funzione restituisce quindi 2, ovvero solo la parte intera del risultato, con ovvia perdita di informazione. Per restituire il quoziente esatto è sufficiente modificare il tipo di ritorno nella definizione.

```
float op(float x, float y) {
        return x/y; }
```