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 D

- **Esercizio 1 (punti 4)**

Dato il seguente programma C:

```
#include <stdio.h>

int mod(int a, int c) {
      if ((a%6) != 0 )
            return mod(a-1, c);
      else
            return ++c;
}

main() {
      int e = 18;
      int f = 3;

      while (e > 0)
      {
            f = mod(e, f);
            printf("%d\n", f);
            e-=6;
      }
}
```

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

- **Esercizio 2 (punti 16)**

a) Si definisca una funzione ricorsiva **`long fatto (long n)`** che calcoli in maniera ricorsiva il fattoriale del parametro n. Si definisca inoltre una funzione iterativa **`float pow (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 H(float e, float f, long n)`** che calcoli il seguente valore: **(punti 6)**

$$\prod_{k=0}^{n} g(n,k)\left(e^{n-k} + f^{k}\right)$$

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, `e`, `f` (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 H() con parametri `e`, `f`, `n` e poi termini. **(punti 6).**

- **Esercizio 3 (punti 5)**

Si consideri la seguente funzione xyz():

```
int xyz(int a, int b) {
      if ((a/b) >= 1)
            return 1 + xyz(a-6, b);
      else
            return (b-1);
}
```

Si scriva il risultato della funzione quando invocata come xyz(21, 5) 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 **{cla,ra,bel,la}**

```
S::= X | X Y
Y::= X W Z | Z X W
X::= cla | ra | bel | la
Z::= ra
W::= la
```

Si dica se la stringa **clarabella** 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 div(int * x, int y) {
      *x = *x  /  y;
      return *x; }

main () { int a=12, b=4;
printf("%d\n", div(&a,b));
printf("%d\n", a); }
```

- **Esercizio 6 (punti 2)**

La seguente funzione vorrebbe calcolare il risultato del prodotto tra il primo parametro ed il secondo. In fase di debugging, però, i valori restituiti da questa funzione sono risultati diversi da quelli attesi, ad esempio quando entrambi i valori dei parametri x e y sono minori di 1. Qual è in tal caso il risultato restituito? Che cosa rende il codice inadatto al calcolo del prodotto di due valori float?

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

Soluzione Compito D

Esercizio 1

Il programma stampa a video la sequenza di interi, ciascuno su una linea diversa:
4 5 6

Il main contiene come prime due istruzioni le dichiarazioni delle variabili e ed f, inizializzate rispettivamente ai valori 18 e 3. 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 f viene assegnato il valore di ritorno della funzione mod(), invocata con parametri e ed f, che valgono rispettivamente 18 e 3 (sono ancora i valori iniziali). La funzione mod() restituisce il risultato della chiamata ricorsiva con parametri a-1, c se l'attuale valore di a non è un multiplo esatto di 6, altrimenti restituisce l'attuale valore di c incrementato di 1. Siccome alla prima invocazione la variabile a vale 18, la condizione di test non è verificata e la funzione restituisce immediatamente il valore di c incrementato (da notare che si tratta di un pre-incremento). Siccome c vale 3, la funzione restituisce il valore 4, che viene assegnato alla variabile f nel main. A questo punto la printf() provvede a stampare il valore di f, che per quanto appena detto è 4. Poi la variabile e viene decrementata di 6 (il suo nuovo valore è 12), e si ricomincia il ciclo.

Alla seconda iterazione la condizione di ingresso nel ciclo è ancora verificata, e le variabili e ed f valgono rispettivamente 12 e 4. Viene ripetuta l'invocazione alla funzione mod(), la quale ritorna il valore c pre-incrementato di 1 (e quindi questa volta vale 5). Il valore 5 viene assegnato alla variabile f, e viene stampato a video dalla printf(). Infine il valore della variabile e viene decrementato di 6. Al termine della seconda iterazione quindi e vale 6 e f vale 5.

Alla terza iterazione la condizione di ingresso nel ciclo è ancora soddisfatta, e alla variabile f viene di nuovo assegnato il valore di ritorno della funzione mod(): siccome mod() viene invocata con parametri attuali a=6 e c=5, e siccome a è un multiplo esatto di 6, mod() restituisce il valore di c pre-incrementato, cioè restituisce 6. Tale valore viene assegnato nel main alla variabile f, che poi viene subito stampata dalla printf(). Infine il valore di e è decrementato di 6 e va a 0.

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

Esercizio 2

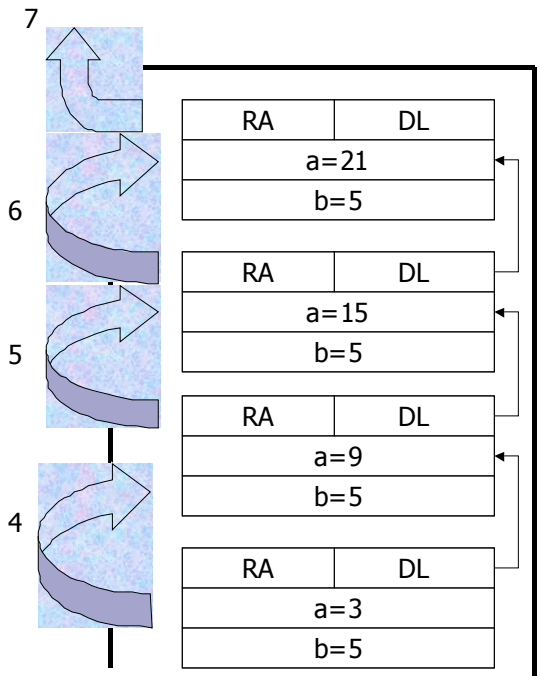
```
long fatto (long n) {
     if (n == 0)
          return 1;
     else       return n * fatto(n-1);
}

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

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

float H(float e, float f, long n) {
     long i;
     float result = 1;
     for (i=0; i<=n; i++) {
          result = result * g(n, i) * potenza(e, n-i) * potenza(f, i);
     }
     return result;
}
```

```
main() {
     float e;
     float f;
     long n;

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

Esercizio 3

Il risultato è 7.



La funzione NON è tail ricorsiva.

Esercizio 4

La stringa è sintatticamente corretta.
Derivazione *left most*:
S → X Y → cla Y → cla Z X W → cla ra X W → cla ra bel W → cla ra bel la

Esercizio 5
3 3
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 di x diviso y. Questo assegnamento modifica anche il valore di a, per quanto detto sopra. La funzione ritorna il valore di x, che vale appunto 3. La seconda istruzione printf() stampa a video il valore attuale di a che, a causa del side-effect nella funzione div(), vale 3.

Esercizio 6
La funzione è sintatticamente corretta ma inadatta a restituire il prodotto fra due float in quanto il tipo del risultato è intero. Nel caso d'esempio, il prodotto di due numeri entrambi minori di 1 produce comunque un risultato che viene troncata al valore intero 0. Per scrivere una funzione che correttamente calcoli il prodotto reale di due numeri reali è sufficiente modificare il prototipo della funzione op() in modo che restituisca un float.