Programmazione II – AA 2003-04
Primo compito di Esonero

A t t e n z i o n e:
Se il primo esercizio, detto di sbarramento, non viene risolto in modo soddisfacente il resto del compito non viene corretto. Avete 1 ora e 45 minuti di tempo.

In Bocca al Lupo!

Esercizio Bonus. Si supponga di ordinare ogni riga di una matrice di interi e successivamente di ordinare ogni colonna. Le righe della matrice rimangono ordinate?

Esercizio di sbarramento. Si scriva una funzione ricorsiva C che, date due liste p e q restituisca 0 se hanno la stessa lunghezza, +1 se p é piú lunga di q e -1 se accade il viceversa.

Esercizio 1. Si scriva una **singola** funzione C ricorsiva per calcolare la somma dei valori di due liste p e q. Le liste contengono interi e sono a puntatori semplici.

Esercizio 2. Cosa stampano i programmi seguenti?

```
void f(int* p) {
  int* q;
  q = p + 1;
  *p = *q;
}
main() {
  int v[3]; v[0] = 10; v[1] = 20; v[2] = 30;
  f(v);
  printf("%d %d %d \n",v[0],v[1],v[2]);
}
```

```
void f(int* p) {
  int* q; int r;
  q = p + 1;
  r = p[2]-1;
  *p = *q + r;
}
main() {
  int v[3];
  v[0] = 100; v[1] = 200; v[2] = 300;
  f(v);
  printf("%d %d %d \n",v[0],v[1],v[2]);
}
```

Esercizio di sbarramento. Si scriva una funzione C ricorsiva per verificare se una lista contiene tutti elementi uguali. La lista contiene numeri interi **maggiori di zero** ed é a puntatori semplici. (Suggerimento: usare lo zero)

Esercizio 1. Cosa stampano i programmi seguenti?

```
void f(int* p) {
  int* q;
  q = p + 2;
  *p = *q;
}
main() {
  int v[3]; v[0] = 50; v[1] = 100; v[2] = 150;
  f(v);
  printf("%d %d %d \n",v[0],v[1],v[2]);
}
```

```
void f(int* p) {
  int* q; int r;
  q = p + 2;
  r = p[1]-3;
  *p = *q - r;
}
main() {
  int v[3];
  v[0] = 100; v[1] = 200; v[2] = 300;
  f(v);
  printf("%d %d %d \n",v[0],v[1],v[2]);
}
```

Esercizio 2. Data una lista $\ell := \ell_1 \to \ell_2 \to \ldots \ell_n$ il suo *prefisso i-esimo* consiste della sottolista $\ell(i) := \ell_1 \to \ell_2 \to \ldots \ell_i$. Vale a dire, il prefisso i-esimo di ℓ sono i suoi primi i elementi. Il *peso* di una lista di interi é la somma dei suoi elementi.

Data una lista di interi ℓ ed un intero x, si scriva una funzione C ricorsiva per verificare se esiste un prefisso di ℓ di peso x.

Dare una stima della complessitá computazionale della vostra funzione.

Esercizio di sbarramento. Data una lista ℓ ed vettore $x[]$ di dimensione nota n, entrambi contenenti numeri interi, si dice che $x[]$ è la *nemesi* di ℓ se $x[]$ contiene gli elementi di ℓ in ordine inverso. Ad esempio se $\ell = 1 \to 2 \to 3 \to 4 \to 5$ allora $x = [5, 4, 3, 2, 1]$ ne è la nemesi. Si scriva una funzione ricorsiva C per verificare se $x[]$ é la nemesi di ℓ. Si puó assumere che essi abbiano lo stesso numero di elementi.

Esercizio 1. Un vettore di interi si dice *palindromo* se letto da sinistra verso destra si ottiene la stessa sequenza di numeri che si otterrebbe leggendolo all'incontrario. Ad esempio,

$$x = [0, 1, 2, 3, 4, 3, 2, 1, 0]$$

é palindromo. Si scriva una funzione C ricorsiva per verificare se un vettore é palindromo.

Esercizio 2. Cosa stampano i programmi seguenti?

```
void f(int* h) {
  int* k;
  k = &h[3];
  *h = *k;
  h++; }

main() {
  int v[5];
  v[0] = 0; v[1] = 1; v[2] = 2; v[3] = 3; v[4] = 4;
  f(v);
  printf("%d %d %d \n",v[0],v[2],v[4]);
}
```

```
void f(int* p) {
  int* q; int r;
  q = p + 3;
  r = q[1];
  *p = *q - r;
}

main() {
  int v[5];
  v[0] = 10; v[1] = 20; v[2] = 30; v[3] = 40; v[4] = 50;
  f(v);
  printf("%d %d %d \n",v[0],v[1],v[2]);
}
```

Esercizio di sbarramento. Dato un vettore $x[]$ di interi dimensione nota n ed un intero k, si scriva una funzione per verificare se $x[]$ contiene un sottovettore di peso k. Un sottovettore é formato da elementi contigui ed il suo peso é dato dalla somma degli elementi in esso contenuti. Ad esempio, se

$$x = [1, 2, 3, 4, 5, 6]$$

si ha che il sottovettore $x[2 \dots 4] = [3, 4, 5]$ ha peso 12.

Esercizio 1. Una lista di interi é una *sega* se gli elementi di posto dispari sono positivi mentre quelli di posto pari sono negativi. Una lista di interi invece é una *controsega* se gli elementi di posto dispari sono negativi mentre quelli di posto pari sono positivi. Una lista é *tagliente* se é una sega oppure una controsega. Si scriva una funzione C per verificare se una lista é tagliente. Usare la ricorsione.

Dare una stima della complessità computazionale della vostra funzione.

Esercizio 2. Cosa stampano i programmi seguenti?

```
void f(int* h) {
  int* k;
  k = (&h[2]) - 1;
  *h = *k;
  k++; }

main() {
  int v[5];
  v[0] = 0; v[1] = 10; v[2] = 20; v[3] = 30; v[4] = 40;
  f(v);
  printf("%d %d %d \n",v[0],v[2],v[4]);
}
```

```
void f(int* p) {
  int* q; int r;
  q = p + 2;
  r = q[2];
  r++;
  *p = *q - r;
}

main() {
  int v[5];
  v[0] = 10; v[1] = 20; v[2] = 30; v[3] = 40; v[4] = 50;
  f(v);
  printf("%d %d %d \n",v[0],v[1],v[2]);
}
```