Esame di Programmazione (mod A) - CdL AIDA

Appello II - Gennaio 2022

Giulio Caravagna (gcaravagna@units.it)

A1 (6 punti). Si consideri la successione:

$$\begin{cases} F_0 = 2 \\ F_n = \frac{1}{2}(1 + \frac{1}{F_{n-1}}) \quad \text{con } n \geq 1 \end{cases}$$

e la quantita $\mathbf{F} = \prod_{i=1}^{m} F_{x_i}$ calcolata a partire da un array di m valori non negativi $\mathbf{x} = [x_1, ...x_m]$. Si scriva un programma C che dato $\mathbf{x}$:

- calcoli $\mathbf{F}$ iterativamente;
- calcoli ogni F_i ricorsivamente.

Esempio: se $x = [1, 2, 0]$ allora $F = F_1 * F_2 * F_0$ dove F_1, F_2 ed F_0 sono ricorsive, il prodotto iterativo.

A2 (6 punti). Dato un insieme di n valori memorizzati in un array a, si definisca una procedura C *ricorsiva* che ordina l'array. Per esempio dato $a = [2, 14, 30, 24, 14]$ restituisce $[2, 14, 14, 24, 30]$.

Suggerimento: Si consiglia di implementare una funzione ausiliaria `int min(int *a, int n, int i)` che calcola l'indice dell'elemento minimo dell' array `a`, di dimensione `n`, restringendosi aell'intervallo $[i, n)$. A quel punto, si osservi che ordinare un array $a_0, a_1, \ldots, a_{n-1}$ equivale a ordinare $a_1, \ldots, a_{n-1}$ *dopo* aver messo in posizione a_0 l'elemento minimo a calcolato tra $[0, n)$.

A3 (6 punti). Si scriva un programma in C che, riceva in input un array a e restituica un array f dove:

- f ha la stessa dimensione di a.
- $f[i] = 1$ se $a[i]$ è presente più di una volta all'interno di a, 0 altriimenti.

Esempio: se $a = [1, 1, 2, 3, 5, 1, 2]$ allora $f = [1, 1, 1, 0, 0, 1, 1]$.

B1 (3 punti). Si consideri il seguente programma di Zibonacci

```c
int zib(int n) {
   if (n == 0)
      return 1;
   if (n <= 2)
      return n;
   if (n%2 == 1 && n >= 3) // odd
      return zib((n-1)/2) + zib((n-1)/2-1) + 1;
   if (n%2 == 0 && n >= 4) // even
      return zib(n/2) + zib(n/2+1) + 1;
}
```

- si calcolini i numeri di Zibonacci fino ad $n = 10$;
- si disegni la memoria massima (in termini di dimensione) raggiunta dalla chiamata `zib(5)`.

B2 (4 punti). Si rappresenti la memoria del programma sottostante ad ogni punto `A`, `B`, `C`.

```
int my_op(int x)
{
  int * y = malloc(size_of(int) * x);
  for(int i = 1; i < x; i++) *(y + i) = i
  int s = 0;
  for(int i = 1; i < x; i++) s += *(y + i) // B
  return(s);
}

int main(void)
{
  int y = 3;
  while(y >= 2)
  {
    int i = my_op(x); // A
    y--;
  } // C
  return(0);
}
```

B3 (5 punti).

Si consideri la seguente classe Python

```
class StrutturaSelettiva():

    def __init__(self, dati, salto):
        self.dati = dati
        self.quale = quale

    def __iter__(self):
        return IteratoreStruttura(self.dati, self.quale)
```

che viene utilizzata in questa maniera

```
print(list(StrutturaSelettiva([1,2,3,4,5], 1)))        # 2,4
print(list(StrutturaSelettiva([1,2,3,4,5], 0)))        # 1,3,5
print(list(StrutturaSelettiva([1,2,3,4,5], 4)))        # 1,3,5
print(list(StrutturaSelettiva([2,3,1,1,5], 1)))        # 3,1
```

ovvero itera sulla lista `dati` mediante un iteratore di classe `IteratoreStruttura`, con la particolarità di mostrare solo gli elementi di indice pari se `quale = 1`, tutti gli altri altrimenti per ogni altro valore di `quale`. Si definisca la classe `IteratoreStruttura`