Esame di Programmazione (mod A)

CDL in Intelligenza Artificiale e Data Analytics.

Giulio Caravagna (gcaravagna@units.it)

I Appello: Gennaio 2021

Istruzioni

L'appello contiene **6** esercizi (A1, A2, A3, B1, B2, B3) da risolvere in 3 ore (tempo massimo).

- Scaricate il file `Appello.zip` da Moodle che contiene un template di 6 cartelle, una per esercizio.
- Le soluzioni devono essere caricate sul portale Moodle alla pagina del corso (in formato `zip`, ricomprimendo le cartelle che avete scaricato come template, dopo aver aggiunto le vostre soluzioni).

Importante. I primi 3 esercizi (A1, A2 e A3) sono considerati di *sbarramento*, le soluzioni devono essere perfette e permettono di raggiungere 18/30 (minimo per superare l'esame). I restanti 3 esercizi (B1, B2 e B3) sono opzionali, e valgono fino al raggiungimento del voto massimo di 30/30.

Risoluzione degli esercizi di sbarramento.

- ogni esercizio va risolto partendo dall'implementazione disponibile nel template. Del template l'unica cosa che dovete modificare è il file `main.c` (come visto a lezione).
- Per esempio, se lavorate su repl.it per risolvere l'esercizio `A1`:
  - create un nuovo repl per C;
  - trascinate la cartella del template `A1` (drag and drop) nel nuovo repl - compariranno tutte le sotto-cartelle ed i files necessari;
  - risolvete l'esercizio modificando `main.c`;
  - avete 3 files di input (`input_1`, `input_2` ed `input_3`) con cui testare il vostro esercizio. I files sono disponibili all'interno della cartella `input` se li volete ispezionare. Il risultato atteso per ciascun input è memorizzato nella cartella `result`. Potete usare questi files per controllare il vostro calcolo.
  - testate la vostra implementazione usando `make test_1` (oppure `make test_2` etc.) per confrontare il risultato da voi calcolato con quello atteso.
  - quando avete finito scaricate la vostra soluzione e sostituitela al template (potete anche sostituire solamente il file `main.c` del template, perchè il resto del template non deve essere modificato).

Esercizi di sbarramento

Es. A1 (6 punti)

Si scriva un programma C che calcola, per un dato $n > 0$ in input, il valore

$$s_n = 2^n \sum_{i=1}^{n} i$$

Il calcolo della potenza e della sommatoria in s_n deve utilizzare *solamente funzioni ricorsive*.

Esempio di calcolo: per $n = 3$ vale $s_n = 48$, per $n = 6$ vale $s_n = 1344$, etc.

Suggerimento. Dovendo le implementazioni essere ricorsive, non è ovviamente ammesso l'uso della funzione `pow`, e di soluzioni analitiche esplicite per il calcolo dei termini di s_n.

Es. A2 (6 punti)

Si consideri un array a in input di dimensione n che memorizza una serie di valori a_1, a_2, ..., a_n in modo sequenziale. Si consideri la seguente formula

$$s_n = \prod_{i=1}^{n-1} (a_i + a_{i+1})$$

Per esempio: se $a = [1, 2, 3, 4, 5, 6]$ allora $s_n = (1+2)(2+3)(3+4)(4+5)(5+6) = 3 * 5 * 7 * 9 * 11 = 10395$;

Si scriva un programma C che calcola s_n in modo ricorsivo.

Es. A3 (6 punti)

Dato un insieme di n valori $x_1, \ldots, x_n$ definiamo la sequente quantità

$$\sigma = \sqrt{\frac{\sum_i (x_i - \mu)^2}{n}}$$

dove $\mu = \frac{1}{n} \sum_i x_i$ è la media campionaria dei valori x_i.

Scrivere un programma C che dato in input un array a di n elementi - i numeri x_i - calcoli il valore di σ associato ad a usando *solamente* funzioni iterative `for` e `while`. Si permette l'utilizzo della funzione `sqrt` per il calcolo della radice quadrata richiesta per σ.

Suggerimento. Si noti che `#include<math.h>`, che serve per avere `sqrt`, è già presente tra gli header del file template per questo esercizio. La radice quadrata si usa come `sqrt(9)=3`. Si presti attenzione alla dichiarazione dei tipi delle variabili al fine di usare `double` per valori non interi.

Esercizi non di sbarramento

Es. B1 (4 punti)

Si consideri questo frammento di codice C

```c
int main(void) {
  int i = 3;
  int a[5];
  int j = i + 1; // P1

  for(int i = 0; i < 4; i++)
  {
    a[i] = i;
    j = i + 1; //P2 (terza iterazione)
  }

  j = i - j + 1; // P3
}
```

Si disegni la memoria del programma al punto P1, P2 (specificatamente alla terza iterazione del ciclo), ed al punto P3 (fine del `main`).

Potete risolvere questo esercizio carta e penna, e caricare uno screenshot del vostro esercizio nella cartella `B1`, oppure consegnare una copia cartacea della vostra soluzione.

Es. B2 (3 punti)

Si consideri questa formula logica

$$\exists i \in \{0, .., 3\} \quad a[a[i]] == 0$$

ed il seguente frammento di programma C, supponendo che la dichiarazione ed inizializzazione dell' array `a` di 4 elementi avvenga correttamente nella porzione di codice mancante `[...]`.

```
[...]

int b[4];

for(int i = 1; i <= 4; i++)
{
  b[i-1] = a[i-1];
  b[i-1] -= a[a[i-1]];

  if(b[i-1] == 0) return(1);
}

return(0);
```

- si fornisca un esempio dei valori del vettore `a` per cui il predicato dovrebbe essere falso;
- si fornisca un esempio dei valori del vettore `a` per cui il predicato dovrebbe essere vero;
- si riscriva la parte di codice mostrata usando tutti questi meccanismi:
  - un ciclo `while`;
  - eliminando il `return` da dentro il corpo del `for`;
  - eliminando l'inutile utilizzo di indici strani dentro il ciclo.

La soluzione può essere scritta dentro un semplice file di testo. Non è necessario fornire un programma completo eseguibile per questo esercizio.

Es. B3 (5 punti)

Ci viene richiesto di implementare, in Python, una gerarchia di classi per lavorare con le funzioni matematiche - generiche $f(x)$ - e ci viene richiesto di implementare un particolare algoritmo per queste funzioni. Come vincolo ci viene data la seguente classe `Function`, che dobbiamo usare obbligatoriamente.

```
class Function:
    "function R -> R"

    def eval(self):
        pass
```

L'algoritmo deve calcolare il valore medio $\hat{f}$ di $f(x)$ su di un intervallo $[a, b]$, con $0 \leq a < b$. La formula per $\hat{f}$ viene definita come segue

$$\hat{f} = \frac{1}{N} \sum_{i=0}^{N-1} f(a + i\delta)$$

dove:

- N sono i sotto-intervalli in cui viene diviso l'intervallo $[a, b]$
- i sotto-intervalli sono tutti della stessa dimensione $\delta > 0$.

Esempio: $[a = 1, b = 3]$, con $N = 2$ otteniamo $\delta = 1$ ed $\hat{f} = 0.5 * [f(1) + f(2)]$.

In questo esercizio si deve:

- modificare la classe `Function` per fornire una funzione `f_hat(self, a, b)` che calcoli $\hat{f}$;
- creare una sottoclasse della nuova `Function` che rappresenti la funzione $f(x) = x^2 + 2x$;
- usare un oggetto della sottoclasse per il calcolo di $\hat{f}$ sull'intervallo $[0, 6]$ con $N = 3$.

Suggerimenti. Si noti che $a + i\delta$ risulta essere la coordinata di inizo dell' i-esimo sotto-intervallo.