Dipartimento di Elettronica e Informazione
Politecnico di Milano
20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (39) 02-2399.3400
Fax (39) 02-2399.3411

Prof. Luca Mottola

Fondamenti di Informatica (ELT)

4 Marzo 2015

Cognome
Nome
Matricola

Note

1. La mancanza dei dati di cui sopra comporta l'annullamento della prova.
2. Restituire **solo** queste pagine. Verranno ignorati fogli aggiuntivi. **Non** si può scrivere con la matita.
3. È proibito l'uso di dispositivi elettronici (computers, calcolatrici, cellulari, macchine fotografiche,…).
4. Qualunque frammento di codice richiesto deve essere scritto in **stampatello**.
5. Sebbene non obbligatorio, si consiglia di inserire commenti nel codice per facilitarne la correzione.
6. Non si può tenere una copia del testo dell'esame quando si lascia l'aula.
7. Un punteggio inferiore a 17 preclude la registrazione del voto complessivo anche se sufficiente.
8. Il tempo a disposizione è di 2 ore.

Domanda 1: Codifica dell'informazione (5 punti)

Codificare due volte in virgola fissa il numero 34.631, supponendo di avere a disposizione 2 byte: nella prima conversione si hanno a disposizione i primi 8 bit per codificare la parte intera ed i secondi 8 bit per la parte frazionaria, nella seconda conversione si utilizzino 6 bit per la parte intera e 10 bit per la parte frazionaria. Non si consideri il segno della parte intera.

Che differenze di approssimazione ci sono tra i due risultati? Dato un numero decimale qualunque, è possibile stabilire a priori quanti bit sarebbero necessari per non avere approssimazioni? Se sì, come è possibile determinare tale numero di bit? Se no, perché non è possibile?

Domanda 2: Macchina di Von Neumann (7 punti)

Scrivere un programma che legga in input una serie di numeri positivi, e controlli che la differenza tra due numeri successivi sia sempre positiva (ovvero, la sequenza di numeri è crescente).
Quando la condizione fallisce, si stampi sul nastro di uscita quanti numeri sono stati letti fino a quel punto, la loro somma, il numero minimo mai letto, e il numero massimo mai letto.

Domanda 3: Linguaggio C (5 punti)

Si consideri il seguente codice in linguaggio C. Per ciascuna funzione e blocco all'interno di ogni funzione, elencare le variabili visibili da quel blocco e il loro tipo. Per identificare funzioni e blocchi, è possibile (ma non obbligatorio) aggiungere opportuni commenti all'interno del codice in corrispondenza di ciascuna funzione e blocco di interesse.

```
int foo(int c);

int main(int argc, const char * argv[]) {

    char a = '0';
    int b=0;

    foo(b++);
    a++;

    if (b>0) {
        int a;
        a++;
    } else {
        a++;
    }

    return 0;
}

int foo(int c){
    int a=1, i;

    for (i=0;i<10;i++){
        float c;
        c=c+1;
        if (a==1) {
            c=c+1;
        }
    }
    return c+a;
}
```

Domanda 4: Ricorsione (4 punti)

John McCarthy (Turing Award, 1971) è stato un informatico che ha contribuito in maniera fondamentale in campi quali l'intelligenza artificiale e i linguaggi di programmazione. Nelle sue lezioni, McCarthy usava la "funzione 91 di McCarthy": una funzione ricorsiva che ritorna sempre 91 per tutti gli ingressi n <= 101. La funzione 91 di McCarthy è definita come segue:

$$M(n) = \begin{cases} n - 10, & \text{se } n > 100 \\ M(M(n + 11)), & \text{se } n \leq 100 \end{cases}$$

Si fornisca un'implementazione ricorsiva in linguaggio C della funzione 91 di McCarthy. Inoltre si indichi, con opportuni commenti nel codice, quali sono i casi basi della ricorsione e quali i passi ricorsivi.

Domanda 5: Strutture dati dinamiche (9 punti)

Si consideri una lista dinamica di interi a singolo puntatore, dove un generico elemento della lista è definito come:

```
typedef struct elemento {
    int a;
    struct elemento* next;
} lista;
```

Si implementi una funzione in linguaggio C che riceve una tale lista in input e produce un'altra lista dove tutte le occorrenze multiple dello stesso intero sono state eliminate. In altre parole, la nuova lista è ottenuta dalla prima eliminando tutti i possibili duplicati. Ad esempio, se la lista data include gli interi {3,4,1,2,1,4,5,4,6}, la nuova lista dovrà includere solo gli interi {3,4,1,2,5,6}.

La funzione che dovete implementare dovrà avere il seguente prototipo:

```
lista* rimuovi_duplicati(lista* data);
```

Nello svolgere l'esercizio, si tenga in considerazione che:

- La nuova lista **non** può essere ottenuta modificando quella data, cioè la nuova lista deve essere completamente separata da quella data;
- E' possibile definire un numero arbitrario di funzioni di supporto, di cui deve essere fornito sia il prototipo sia la corrispondente implementazione;
- **Non** è possibile fare uso di variabili globali.