Politecnico di Milano

Dipartimento di Elettronica e Informazione

Informatica A – a.a. 2013/2014 – 19/02/2014

Cognome	_____	**Matricola**	_____
Nome	_____	**Firma dello studente**	_____

Istruzioni

- Non separate questi fogli. Scrivete la soluzione **solo sui fogli distribuiti**, utilizzando il retro delle pagine in caso di necessità. **Cancellate le parti di brutta** (o ripudiate) con un tratto di **penna**.
- Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
- **È possibile scrivere a matita** (e non occorre ricalcare al momento della consegna!).
- È **vietato** utilizzare **calcolatrici, telefoni o pc**. Chi tenti di farlo vedrà **annullata** la sua prova.
- **Non è ammessa la consultazione di libri** e **appunti**.
- Qualsiasi **tentativo** di comunicare con altri studenti comporta **l'espulsione** dall'aula.
- È possibile **ritirarsi senza penalità**.
- Non è possibile lasciare l'aula conservando il tema della prova in corso.
- Tempo a disposizione: **2h30m**

Valore indicativo degli esercizi, voti parziali e voto finale:

Esercizio 1 (4 punti)

Esercizio 2 (4 punti)

Esercizio 3 (5 punti)

Esercizio 4 (7 punti)

Esercizio 5 (8 punti)

Totale: (completo 28 punti)

Esercizio 1 - Algebra di Boole, Aritmetica Binaria, Codifica delle Informazioni (4 punti)

(a) Si costruisca la tabella di verità della seguente espressione booleana in tre variabili, badando alla precedenza tra gli operatori logici. Eventualmente si aggiungano le parentesi. Non si accetteranno soluzioni senza il procedimento. (1 punto)

(A or not B and C) or (not A or B) and not C

(b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri A = 39 e B = –72, li si converta, se ne calcolino la somma (A+B) e la differenza (A–B) in complemento a due e si indichi se si genera riporto sulla colonna dei bit più significativi e se si verifica overflow. Non si accetteranno soluzioni senza il procedimento. (2 punti)

(c) Si converta il numero 99.65625 in virgola fissa e in virgola mobile IEEE 754 (sapendo che 1/2 = 0.5, 1/4 = 0.25, 1/8 = 0.125, 1/16 = 0.0625, 1/32 = 0.03125, 1/64 = 0.015625, e 1/128 = 0.0078125). Non si accetteranno soluzioni senza il procedimento. (1 punto)

Esercizio 2 - Domanda di teoria (4 punti)

Spiegare che cosa si intende per iterazione e per ricorsione. Quando è necessario, o conveniente, usarle? In generale è più efficiente l'iterazione o la ricorsione? Implementare una semplice funzione di esempio sia in versione iterativa che in versione ricorsiva.

Esercizio 3 - Matlab (5 punti)

Scrivere il codice Matlab che restituisca i valori richiesti. Attenersi al numero di righe di codice indicato. L'eventuale utilizzo di più righe comporterà una minore valutazione dell'esercizio.

Data la matrice:

```
A =

    32    12    -1    42
   -21    17   -18   -53
   -27    11    10    25
```

1. Creare la matrice A (max 1 riga) (0.5 punti):

2. Chiedere un numero all'utente e stampare solo i valori di A minori del numero inserito (max 2 righe) (1 punto)

3. Inserire una colonna contenente valori casuali da 3 a 10 dopo la prima colonna (max 2 righe) (1 punto):

4. Creare una matrice B contenente solo i valori maggiori della media dei soli valori positivi (max 1 riga) (1.5 punti):

5. Creare una funzione che restituisca una matrice che corrisponda alla sottomatrice composta dalla prima e terza colonna della matrice A (max 2 righe) (1 punto)

Esercizio 4 - Programmazione C (Ricorsione) (7 punti)

Implementare la seguente funzione **RICORSIVA**

void stampaCombinazioni(char stringa[], int indice)

che, presa una stringa composta solamente di caratteri '0', '1' e 'x', stampi a video tutti i possibili numeri binari che hanno al posto delle 'x' o il carattere '0' o il carattere '1'. Il parametro **indice** rappresenti la posizione a partire dalla quale la funzione deve "prende in considerazione" la stringa.

Ad esempio, dato il seguente codice

```
int main(int argc, const char * argv[])
{
    char stringa[20];
    scanf("%s",stringa);
    stampaCombinazioni(stringa,0);
    return 0;
}
```

e inserendo come stringa "1x0x", la funzione stamperà a video "1000", "1001", "1100" e "1101".

Esercizio 5 - Programmazione in C (Liste Dinamiche) (8 punti)

Si ipotizzi di avere le seguenti strutture dati per la gestione di una lista di persone.

```
typedef struct p {
  char nome[20];
  char cognome[20];
  int eta;
  struct p *next;
} Persona;

typedef Persona *ptrPersona;
```

1. Date le seguenti strutture dati

   ```
   typedef struct pos {
     int dato;
     struct pos *next;
   } Posizione;

   typedef Posizione *ptrPosizione;
   ```

   si implementi la seguente funzione **RICORSIVA**.

   void listaPosizioni(ptrPersona testa, char cognome[20], ptrPosizione *testaPosizione, int posizione);

   Dati in ingresso la testa di una lista di persone (**testa)** e un **cognome**, la funzione crei una seconda lista in **testaPosizione** che contiene le posizioni della lista originale in cui ci sono persone con quel cognome.

2. Date le seguenti strutture dati

```
typedef struct f {
  char cognome[20];
  ptrPersona membro;
  struct f *next;
} Famiglia;

typedef Famiglia *ptrFamiglia;
```

si implementi la seguente funzione **NON RICORSIVA**.

ptrFamiglia listaDiListe(ptrPersona testa);

Dati in ingresso la testa di una lista (**testa**), la funzione crei una struttura dati di tipo lista di liste. La lista di liste deve consistere in una riorganizzazione della lista originale in cui tutti i membri della stessa famiglia (ovvero le persone che hanno lo stesso cognome) sono riuniti tra di loro. Si prenda ad esempio la figura sottostante, in cui dalla prima lista viene generata la lista di liste.

NB. La lista originale NON deve essere modificata. La lista di liste deve consistere di copie delle strutture dati di tipo Persona che sono presenti nella lista originale.