Politecnico di Milano

Dipartimento di Elettronica e Informazione

Informatica A – a.a. 2010/2011 – 07/07/2011

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: **2h15m**

Valore indicativo degli esercizi, voti parziali e voto finale:

Esercizio 1 (4 punti)

Esercizio 2 (5 punti)

Esercizio 3 (6 punti)

Esercizio 4 (5 punti)

Esercizio 5 (8 punti)

Totale: (completo 28 punti)

Esercizio 1 - Algebra di Boole, Aritmetica Binaria, Codifica delle Informazioni

(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 (1 punto).

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

(b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri $A = 35_{dec}$ e $B = -72_{dec}$, 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 (2 punti).

(c) Si converta il numero 83.0625 in virgola fissa e in virgola mobile ($r = m \times 2^n$, con m e n codificati in binario, 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) (1 punto)

Esercizio 2 - Interpretazione del codice C

Si consideri il seguente codice:

```c
#include <stdio.h>

void MM (int a1[][2], int a2[][3], int a3[][3]);
void PM (int ar[][3], int n);

int main(void)
{
    int A[3][2] = {{a00, a01},
        {a10, a11},
        {a20, a21}},
    B[2][3] = {{b00, b01, b02},
        {b10, b11, b12}},
    C[3][3] = {{0, 0, 0},
        {0, 0, 0,},
        {0,0,0}};

    MM(A, B, C);
    PM(C, 3);

    return 0;
}


void MM(int a1[][2], int a2[][3], int a3[][3])
{
    int i = 0;
    int j = 0;
    int k = 0;
    for(i = 0; i < 3; i++)
        for( j = 0; j < 3; j++)
            for( k = 0; k < 2; k++)
                a3[i][j] +=  a1[i][k] * a2[k][j];
}

void PM (int ar[][3], int n)
{
    int i = 0;
    int j = 0;
    for(i = 0; i < n; i++)
    {
        for( j = 0; j < 3; j++)
            printf("%4d", ar[i][j]);
        printf("\n");
    }
}
```

Si sostituisca ad a00, a01, a10, a11, a20 e a21 le 6 cifre della propria matricola. Si sotituisca a b00, b01, b02, b10, b11, b12 le cifre della propria data di nascita secondo lo schema gg/mm/aa.

1. Spiegare cosa fanno le funzioni "MM" e "PM".
2. Indicare chiaramente cosa viene stampato a video eseguendo la funzione "main".

Esercizio 3 - DB

Sia data la seguente basi di dati per la gestione di un servizio di immagazzinamento di file online:

UTENTE (Id_utente, Nome, Cognome, Username, Password)

FILE (Id_file, Id_utente, Filename, Dimensione, Data_upload)

1. Scrivere la query SQL che estrae il numero di utenti che utilizzano il proprio cognome come username.
2. Scrivere la query SQL che estrae per ciascuna persona l'identificativo, il nome, il cognome e la quantità di dati immagazzinata nel sistema. Si considerino solo quegli utenti che occupano almeno 1 Giga di memoria.
3. Scrivere la query SQL che estrae per ciascuna persona l'identificativo, il nome, il cognome e il numero di file immagazzinati nel sistema. Si considerino solo quegli utenti che hanno fatto upload nel mese di Luglio 2011.

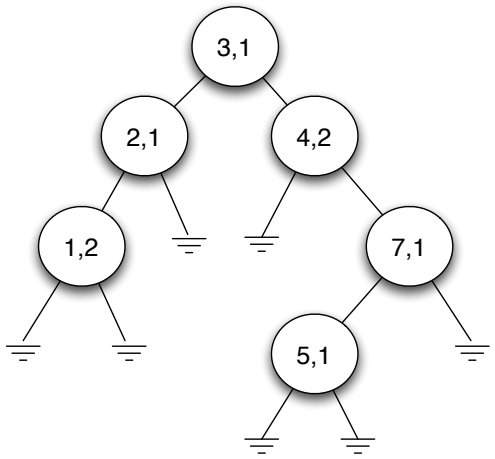
Esercizio 4 - Domanda di Teoria

Perché i puntatori sono importanti in C? Descrivere i vari casi in cui il loro uso è importante e riportare per ciascun caso un esempio concreto.

Esercizio 5 - Ricorsione in C

Si consideri il seguente codice:

```
typedef struct nodo {
     int valore;
     int occorrenze;
     struct nodo *left;
     struct nodo *right;
} node;
typedef node *ptrNode;

ptrNode addToTree(int n, ptrNode t);
void printTreePreOrder(ptrNode t);

int main (int argc, const char * argv[])
{
    ptrNode t = NULL;
    int values[8] = {3, 2, 4, 4, 1, 1, 7, 5};
    int i;

    for (i=0; i<8; i++) {
        t = addToTree(values[i], t);
    }

    printTreePreOrder(t);

}
```



Il codice prende un array di 8 interi e ne produce una visualizzazione compatta basandosi su un albero. L'albero viene creato usando la funzione "addToTree", e viene visualizzato secondo una visita preordine mediante la funzione "printTreePreOrder".

L'albero è fatto nel seguente modo. Ogni nodo contiene un valore letto dall'array e il numero di occorrenze di quel valore nell'array. Se un valore è ripetuto più volte nell'array, esso viene rappresentato da un solo nodo nell'albero.

Inoltre, preso un nodo qualsiasi dell'albero, tutti i nodi del sottoalbero sinistro hanno un valore più piccolo del valore presente nel nodo stesso, mentre tutti i nodi del sottoalbero destro hanno un valore più grande del valore presente nel nodo stesso.

In figura viene riportato l'albero creato a partire dall'array di numeri "values" presente nella funzione "main". Si noti che l'albero non è bilanciato. **NON** è necessario creare un albero bilanciato.

1. Implementare la funzione "printTreePreOrder" in maniera **RICORSIVA**. La visita preordine dell'albero in figura produce la stampa (3,1) (2,1) (1,2) (4,2) (7,1) (5,1)
2. Implementare la funzione "addToTree" in maniera **RICORSIVA**. La funzione deve incrementare il numero di occorrenze del valore "n" (passato come parametro) se esiste già un nodo per quel valore nell'albero. Se il nodo non esiste, la funzione deve creare un nuovo nodo, impostare il numero di occorrenze a 1, e posizionarlo nel posto opportuno dell'albero.