Politecnico di Milano

Dipartimento di Elettronica e Informazione

Informatica A – a.a. 2010/2011 – 16/09/2011

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 (5 punti)

Esercizio 4 (5 punti)

Esercizio 5 (9 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).

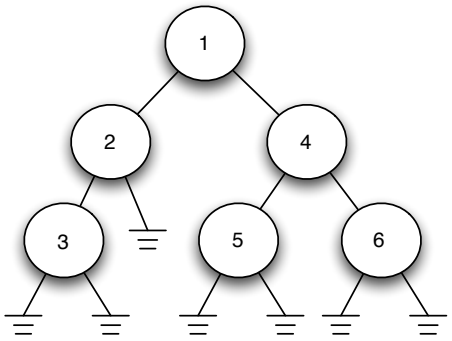
(A and not B) or (not A or C)

(b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri $A = 28_{dec}$ e $B = -56_{dec}$, 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 14.0375 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:




```
typedef struct nodoAlbero {
    int dato;
    struct nodoAlbero *left;
    struct nodoAlbero *right;
} treeNode;
typedef treeNode *ptrTreeNode;

typedef struct nodo {
    ptrTreeNode foglia;
    struct nodo *next;
} node;
typedef node *ptrNode;

void cl(ptrTreeNode t, ptrNode *l);
void p(ptrNode p);

int main ()
{
    ptrNode t;
    ptrTreeNode root = (ptrTreeNode)malloc(sizeof(treeNode));

     //codice che crea l'albero mostrato in figura...

    cl(root, &t);
    p(t);

}

void cl(ptrTreeNode t, ptrNode *l) {
     ptrNode temp;
     if(t==NULL)
       return;
     if(t->left==NULL && t->right==NULL) {
       temp=*l;
       (*l=(ptrNode)malloc(sizeof(node));
       (*l)->foglia=t;
       (*l)->next=temp;
     }
     if(t->left!=NULL)
       cl(t->left, l);
     if(t->right!=NULL)
       cl(t->right, l);
}

void p(ptrNode p) {
    ptrNode temp = p;
    while (temp != NULL) {
        printf ("%d ", temp->foglia->dato);
        temp = temp->next;
    }
}
```

Spiegare in maniera dettagliata che cosa fanno le due funzioni **void cl (ptrTreeNode t, ptrNode *l)** e **void p(ptrNode p)** e spiegare "come" lo fanno.

Indicare infine che cosa viene stampato a video quando viene eseguito il main.

Esercizio 3 - DB

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

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

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

DIZIONARIO (Password)

Si ipotizzi che la tabella DIZIONARIO venga utilizzato per mantenere password ritenute "deboli", ovvero password che devono essere sconsigliate all'utente nel caso le dovesse selezionare.

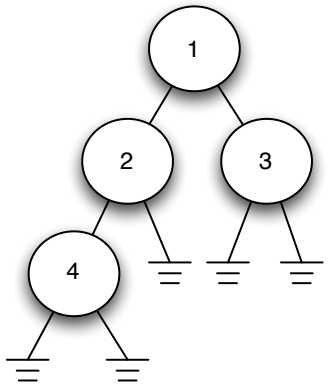
1. Scrivere la query SQL che estrae i cognomi e nomi degli utenti che fanno uso di password deboli.
2. Scrivere la query SQL che estrae per ciascuna persona l'identificativo, il nome, il cognome e il numero di immagini immagazzinati nel sistema. Si considerino solo quegli utenti che hanno caricato nel sistema almeno una volta un file di dimensione superiore ai 10 mega.

Esercizio 4 - Domanda di Teoria

Spiegare cosa si intende per procedura ricorsiva. Perché la ricorsione è uno strumento importante nella programmazione?

Esercizio 5 - Ricorsione in C

Si consideri le seguenti strutture dati e funzioni:




```
typedef struct treeNodo {
    int dato;
    struct treeNodo *left;
    struct treeNodo *right;
} treeNode;
typedef treeNode * ptrTreeNode;

typedef struct nodo {
    ptrTreeNode ptrInfo;
    struct nodo * next;
} node;
typedef node *ptrNode;

void pushQueue(ptrNode *root, ptrTreeNode ptr);
ptrTreeNode popQueue(ptrNode *root);
void printTree(ptrTreeNode root);

int main ()
{
     ptrTreeNode root = (ptrTreeNode)malloc(sizeof(treeNode));

    root->dato = 1;
    root->left = (ptrTreeNode)malloc(sizeof(treeNode));
    root->right = (ptrTreeNode)malloc(sizeof(treeNode));
    root->left->dato=2;
    root->right->dato = 3;
    root->right->left= NULL;
    root->right->right =NULL;
    root->left->right=NULL;
    root->left->left = (ptrTreeNode)malloc(sizeof(treeNode));
    root->left->left->dato = 4;
    root->left->left->left = NULL;
    root->left->left->right = NULL;

    printTree(root);

}
```

Il codice crea l'albero binario mostrato in figura e poi stampa i suoi elementi secondo una visita breadth-first. La visita breadth-first esaurisce tutti i nodi di ciascun livello, da sinistra verso destra, prima di scendere al livello successivo. Nel caso dell'esempio viene stampato: 1 -> 2 -> 3 -> 4.

La stampa **printTree** viene fatta utilizzando una struttura dati di supporto. In particolare viene utilizzata una coda che viene implementata usando la struttura dati più semplice **node**. La funzione inserisce nella coda l'elemento radice. Fin tanto che la coda non è vuota, toglie il primo elemento, lo stampa, e poi, se esistono, inserisce in coda i figli dell'elemento appena stampato. Nel nostro esempio prima inserisce 1 producendo la coda {1}, poi toglie l'elemento, lo stampa, e inserisce i suoi figli, producendo così la coda {2, 3}. A questo punto toglie il 2, lo stampa e inserisce i suoi figli, producendo la coda {3, 4}. Procede così fino a quando la coda non è vuota.

1. Implementare la funzione **void pushQueue (ptrNode *root, ptrTreeNode ptr)** che inserisce un nuovo elemento nella coda. L'elemento da inserire è il puntatore al treeNode dell'albero che si vuole trattare.
2. Implementare la funzione **ptrTreeNode popQueue(ptrNode *root)** che restituisce e toglie il primo elemento della coda.
3. Implementare la funzione **void printTree (ptrTreeNode root)** che utilizza le due funzioni di cui sopra.