Politecnico di Milano

Dipartimento di Elettronica e Informazione

Informatica A – a.a. 2011/2012 – 07/02/2011

Cognome _____ **Matricola**

Nome _____ **Firma**

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: **2h**

Valore indicativo degli esercizi, voti parziali e voto finale:

Esame completo: esercizi 1-5

Esercizio 1 (completo 4 punti)

Esercizio 2 (completo 4 punti)

II Compitino: esercizi 3-5

Esercizio 3 (completo 6 punti || II compitino 5 punti)

Esercizio 4 (completo 6 punti || II compitino 4 punti)

Esercizio 5 (completo 8 punti || II compitino 5 punti)

Totale: (completo 28 punti | II compitino 14 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 or B and not C) or (A and C) or not B**

(b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri $A = 25_{dec}$ e $B = -62_{dec}$, 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 47.5625 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 - Domanda di Teoria

Illustrare la differenza tra il passaggio di parametri per valore e il passaggio di parametri per indirizzo.

Descrivere come è possibile realizzare l'effetto di un passaggio per indirizzo nel linguaggio C.

In quali situazioni è preferibile l'utilizzo del passaggio per indirizzo?

Esercizio 3 - Basi di dati

Sia data la seguente base di dati per incidenti stradali.

PolizzaAuto (NumeroPolizza, DataInizio, DataScadenza, Assicurato, Auto, ClasseMerito, Premio)

Incidente (Id_Incidente, Data, NumeroPolizzaColpevole, NumeroPolizzaNonColpevole, CostoPerAgenzia)

Auto (Id, Casa, Modello, Cilindrata, Valore)

1. Scrivere la query SQL che restituisce il numero di polizze auto in classe di merito 15 e la somma dei loro premi.
2. Scrivere la query SQL che restituisce il Codice Fiscale delle persone che hanno più di una polizza auto.
3. Scrivere la query SQL che restituisce per ogni casa automobilistica il numero di incidenti in cui il colpevole guidava un'auto di quella casa.

Esercizio 4 - Programmazione C

Un numero si dice perfetto quando è uguale alla somma dei suoi divisori propri. Ad esempio, il numero 28, divisibile per 1, 2, 4, 7, 14 è un numero perfetto (28 = 1 + 2 + 4 + 7 + 14).

1. Implementare la funzione **int isPerfetto(int n)** che restituisce 1 se n è un numero perfetto, o 0 altrimenti.

2. Implementare un programma che, utilizzando **isPerfetto**, stampi a video tutti i numeri perfetti da 1 a MAX=32000.

Esercizio 5 - Strutture dati dinamiche

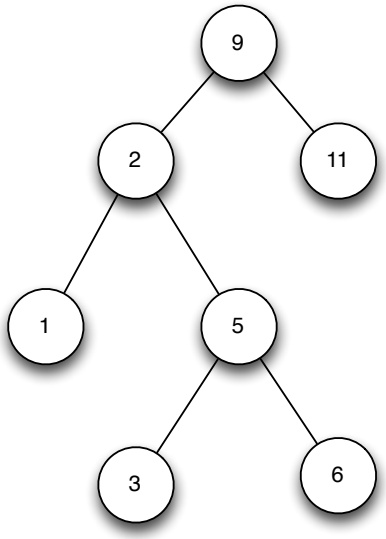
```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 7

typedef struct nodo {
    int num;
    struct nodo *left;
    struct nodo *right;
} node;

typedef node *ptrNode;

ptrNode ordina(int array[MAX]);
ptrNode insert(ptrNode testa, int elem);
void print(ptrNode testa);

int main ()
{
    int array[MAX] = {9,11,2,5,3,6,1};
    ptrNode ordinato;
    ordinato = ordina(array);
    print(ordinato);
    return 0;
}
```



Il codice utilizza un albero binario per ordinare un array di interi. Come si vede dalla figura, viene creato un albero che ha come radice il primo elemento dell'array. Gli altri elementi vengono poi inseriti, di volta in volta, o nel sotto-albero sinistro o in quello destro, a seconda che siano inferiori o maggiori della radice.

Esempio:

inserimento del numero 9 -> viene creata la radice dell'albero
inserimento del numero 11 -> siccome 11 è maggiore di 9 viene creato il sotto-albero destro
inserimento del numero 2 -> siccome 2 è minore di 9 viene creato il sotto-albero sinistro
inserimento del numero 5 -> siccome 5 è minore di 9 si guarda a sinistra. siccome 5 è maggiore di 2 si crea un sotto-albero destro rispetto al nodo 2.
e via dicendo...

1. Implementare la funzione RICORSIVA **ptrNode insert(ptrNode testa, int elem)** che prende la testa di un albero e un intero, inserisce l'intero nell'albero secondo la strategia spiegata sopra, e restituisce la nuova testa dell'albero.

2. Implementare la funzione **ptrNode ordina(int array[MAX])** che prende l'array e restituisce la testa dell'albero ordinato. Questa funzione dovrebbe fare uso della funzione insert.

3. Implementare la funzione RICORSIVA **void print(ptrNode testa)** che riceva la testa di un albero e stampa i suoi elementi secondo una visita post-ordine sinistro, ovvero stampa prima il sotto-albero sinistro, poi la radice, e infine il sotto-albero destro. L'effetto sarà la stampa degli elementi dell'array originale in ordine crescente.