Politecnico di Milano

Dipartimento di Elettronica e Informazione

Informatica A – a.a. 2010/2011 – 12/11/2010

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:

Esercizio 1 (x punti) _____

Esercizio 2 (x punti) _____

Esercizio 3 (x punti) _____

Esercizio 4 (x punti) _____

Esercizio 5 (x punti) _____

Totale: (x punti) _____

Esercizio 1 - Algebra di Boole, Aritmetica Binaria, Codifica delle Informazioni (x 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 (1 punto).

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

(b) Si stabilisca il minimo numero di bit sufficiente a rappresentare in complemento a due i numeri $A = 46_{dec}$ e $B = -28_{dec}$, 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 12.5625 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 - Architettura del calcolatore (x punti)

Si descriva brevemente le differenze tra la memoria centrale e la memoria di massa di un calcolatore. Si spieghi inoltre perché un calcolatore necessita di entrambe.

Esercizio 3 - Interpretazione del codice (x punti)

Completare il seguente codice con la propria matricola. Indicare e spiegare cosa viene stampato a video.

```c
#define N 6

int main () {

    int matricola = _____ ;
    int i;
    int div;
    int parz;
    char ch;

    div = 10;
    for (i=0; i<N; i++) {
          parz = (matricola % div) / (div / 10);
          printf("parziale: %d", parz);
          if (parz % 2 == 0) {
               ch = '-';
          }
          else {
               ch = 'a' + parz;
          }
          printf(" - codifica: %c\n", ch);
          div *=10;
    }

    return 0;

}
```

Esercizio 4 - Puntatori (x punti)

4.1 Il seguente codice fa parte di un'applicazione di vendita di musica online. Indicare e spiegare cosa viene stampato a video.

```
#include <stdio.h>
#include <string.h>

typedef struct {
        char username[20];
        int credit;
} account;

typedef account *ptrAccount;

void addCredit1(account a, int amount);
void addCredit2(ptrAccount ptr, int amount);

void stampaAccount(account a);

int main () {
        account a1;
        ptrAccount ptr;

        strcpy(a1.username, "mickey");
        a1.credit=10000;

        //Punto A//
        printf("Stampa di a1.\n");
        stampaAccount(a1);
        printf("\n");

        //Punto B//
        printf("Esecuzione di addCredit1.\n");
        printf("Stampa di a1.\n");
        addCredit1(a1, 5000);
        stampaAccount(a1);
        printf("\n");

        //Punto C//
        printf("Esecuzione di addCredit2.\n");
        addCredit2(&a1, 5000);
        printf("Stampa di a1.\n");
        stampaAccount(a1);
        printf("\n");

        return 0;
}
```

```
void addCredit1(account a, int amount) {
        a.credit = a.credit + amount;
}

void addCredit2(ptrAccount ptr, int amount) {
        (*ptr).credit = (*ptr).credit + amount;
}


void stampaAccount(account a){
        printf("Username: %s\n", a.username);
        printf("Credit: %d\n", a.credit);
}
```

4.2 Se volessimo estendere il tipo account con un elenco di tutti gli acquisti fatti dall'utente, il dato potrebbe diventare molto grosso dopo non molto tempo. In questo caso, quale versione della funzione **addCredit** sarebbe da preferire e perché?

Risposta esercizio 4.1

Risposta esercizio 4.2

Esercizio 5 - Algoritmi (x punti)

Nel gioco del Forza Quattro si utilizza una matrice di caratteri 6x7 per tenere traccia del tabellone. Ogni posizione può contenere il carattere '**_**' per indicare un posto vuoto, il carattere '**R**' per indicare un pezzo rosso, oppure il carattere '**G**' per indicare un pezzo giallo.

Implementare la funzione

int vince(char colore, char tabellone[6][7])

La funzione prende in ingresso un colore ('R' o 'G') e la matrice di caratteri; restituisce 1 se il giocatore di quel colore, data la situazione del tabellone, ha vinto la partita e 0 se la partita è ancora aperta. Un giocatore vince la partita se almeno 4 pezzi del suo colore sono posti vicini, verticalmente, orizzontalmente, o in diagonale.

Si faccia uso delle seguenti funzioni **(senza implementarle)**.

1. int contaVerticale (char colore, int x, int y, int direzione, char m[6][7]) - La funzione conta quante volte è presente, consecutivamente, il carattere "colore" all'interno della matrice m, a partire dalla posizione (x,y). Il conteggio viene fatto verticalmente, o verso l'alto o verso il basso, a seconda del valore di "direzione". Se direzione vale 1 conta verso l'alto, se vale -1 conta verso il basso.
1. int contaOrizzontale (char colore, int x, int y, int direzione, char m[6][7]) - La funzione conta quante volte è presente, consecutivamente, il carattere "colore" all'interno della matrice m, a partire dalla posizione (x,y). Il conteggio viene fatto orizzontalmente, o verso destra o verso sinistra, a seconda del valore di "direzione". Se direzione vale 1 conta verso destra, se vale -1 conta verso sinistra.
2. int contaDiagonale (char colore, int x, int y, int direzione, char m[6][7]) - La funzione conta quante volte è presente, consecutivamente, il carattere "colore" all'interno della matrice m, a partire dalla posizione (x,y). Il conteggio viene fatto in diagonale, o verso l'alto e a destra, o verso l'alto e a sinistra, o verso il basso e a destra, o verso il basso e a sinistra, a seconda del valore di "direzione". Se direzione vale 1 conta verso l'alto e a destra, se vale 2 conta verso l'alto e a sinistra, se vale 3 conta verso il basso e a destra, mentre se vale 4 conta verso il basso e a sinistra.

Una volta terminata l'implementazione della funzione **vince**, si implementi la funzione **contaVerticale.**

NB. Si faccia attenzione ai casi in cui i parametri 'x' e 'y' ricevuti siano "vicini" al bordo del tabellone.