Informatica B – a.a. 08/09 – Appello – 20/7/2009

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 ricalcare al momento della consegna!).
- È **vietato** utilizzare **calcolatrici** o **telefoni**. Chi tenti di farlo vedrà **annullata** la sua prova.
- È ammessa la consultazione di **libri** e **appunti**, purché con pacata discrezione e senza disturbare.
- 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: **2 h**

Valore degli esercizi, voti parziali e voto finale:

Esercizio 1 (4 punti) _____

Esercizio 2 (10 punti) _____

Esercizio 3 (10 punti) _____

Esercizio 4 (8 punti) _____

Totale: (32 punti) _____

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

Si costruisca la tabella di verità della seguente espressione booleana in tre variabili.

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

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

Esercizio 2 (10 punti)

Una matrice Matlab contiene numeri interi. Si vuole progettare una funzione che ricevendo la matrice e un array di numeri interi che rappresenta una sequenza, cerchi tale sequenza all'interno della matrice. La sequenza può essere disposta, nella matrice, verticalmente dall'alto verso il basso od orizzontalmente, da sinistra verso destra. La funzione deve avere la seguente intestazione:

function [riga, col, dir] = cercaInizioSequenza(matrice, seq)

se la sequenza è presente nella matrice allora *riga* e *col* indicano gli indici di riga e di colonna del suo primo elemento, mentre *dir* viene posto uguale al carattere 'v' se la sequenza è disposta verticalmente, 'o' se orizzontalmente (se la sequenza è presente ripetuta in più posizioni, i valori restituiti possono essere quelli relativi a una qualsiasi delle ripetizioni); se la sequenza non è presente, *riga* e *col* valgono entrambi 0 e *dir* vale 'n'.

1. Per codificare la funzione in questione, si sviluppano prima le due seguenti funzioni ausiliarie

   function [pres] = verificaSeqOrizzontaleDaPosizione (matrice, seq, riga, col)
   function [pres] = verificaSeqVerticaleDaPosizione (matrice, seq, riga, col)

   che ricercano la sequenza nella matrice a partire da una posizione d'inizio precisa, in direzione orizzontale o verticale: *riga* e *col* sono il punto di inizio; il risultato *pres* vale 1 se la sequenza è presente, 0 altrimenti.

2. Successivamente si utilizzano queste due funzioni per codificare le due seguenti

   function [riga, col] cercaInizioSeqOrizzontale = (matrice, seq)
   function [riga, col] cercaInizioSeqVerticale = (matrice, seq)

   che ricercano la sequenza in tutta la matrice con disposizione orizzontale e verticale, restituendo in riga e col le coordinate del punto d'inizio, se la sequenza viene trovata, o il valore 0 altrimenti.

3. Infine si codifica la funzione cercaInizioSequenza facendo uso delle due precedenti.

Ai fini dell'esame, per questioni di tempo, si sviluppino solo le seguenti funzioni, ipotizzando che le altre due siano già state sviluppate:

- function [pres] = verificaSeqOrizzontaleDaPosizione (matrice, seq, riga, col)
- function [riga, col] cercaInizioSeqOrizzontale = (matrice, seq)
- function [riga, col, dir] = cercaInizioSequenza(matrice, seq)

Soluzione

```
function  [pres] = verificaSeqOrizzontaleDaPosizione (matrice, seq, riga, col)
len=length(seq);
pr=matrice(riga,col:col+len-1);
pres=all(pr==seq);

function  [pres] = verificaSeqVerticaleDaPosizione (matrice, seq, riga, col)
len=length(seq);
pr=matrice(riga:riga+len-1,col);
pres=all(pr==seq');

function  [riga, col] = cercaInizioSeqOrizzontale (matrice, seq)
len = length(seq);
[R,C] = size(matrice);
for r = 1 : R
  for c = 1: C-len+1
    pres = verificaSeqOrizzontaleDaPosizione(matrice, seq, r, c);
    if  pres
      riga=r;
      col=c;
       return;
    end;
  end;
end;
riga=0;
col=0;
```

```
function  [riga, col] = cercaInizioSeqVerticale (matrice, seq)
len = length(seq);
[R,C] = size(matrice);
for c = 1 : C
  for r = 1: R-len+1
    pres = verificaSeqVerticaleDaPosizione(matrice, seq, r, c);
    if  pres
      riga=r;
      col=c;
       return;
    end;
  end;
end;
riga=0;
col=0;

function  [riga, col, dir] = cercaInizioSequenza(matrice, seq)
  [ro, co] = cercaInizioSeqOrizzontale(matrice, seq);
  if ro ~= 0
    riga = ro;
    col = co;
    dir = 'o';
    return;
  end;
  [rv, cv] = cercaInizioSeqVerticale(matrice, seq);
  if rv ~= 0
    riga = rv;
    col = cv;
    dir = 'v';
    return;
  end;
  riga=0;
  col=0;
  dir='n';
```

Esercizio 3 (10 punti)

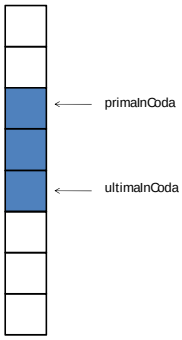
In un'officina informatizzata ogni auto viene registrata all'ingresso, poi, in funzione del motivo per cui si trova nell'officina, o passa al reparto diagnosi in cui viene esaminata per individuare il guasto oppure va direttamente al reparto manutenzione ordinaria in cui si effettuano le normali operazioni di cambio di olio, filtri, ecc. Dal reparto diagnosi l'auto passa al reparto manutenzione straordinaria in cui il guasto viene riparato. Le informazioni sul tipo di intervento effettuato vengono infine archiviate insieme alle altre informazioni sull'auto in modo da poter essere riutilizzate in caso di necessità.

a) Si definisca in C un tipo di struttura dati adeguato per rappresentare tutte le informazioni sull'auto di cui il sistema informativo dell'officina dovrebbe tenere traccia (si supponga per semplicità che il sistema debba tenere traccia delle ultime 100 manutenzioni effettuate sull'auto).

b) Supponendo che il tipo definito al punto precedente si chiami auto, e supponendo che la coda delle auto in attesa nel reparto diagnosi sia definita nel modo seguente:

```
typedef struct {
  auto elenco[20];
  int primaInCoda; /* indica la posizione della prima auto in coda */
  int ultimaInCoda; /* indica la posizione dell'ultima auto in coda */
} codaAuto;
codaAuto lista;
```

Si scriva il frammento di codice C che aggiorna opportunamente il campo primaInCoda in modo tale che ogni volta si esamini una nuova auto secondo una politica FIFO (First In First Out, primo arrivato primo servito).
Il disegno seguente mostra una possibile situazione dell'array elenco (per semplicità il disegno mostra un array di sole 8 posizioni). Le celle in grigio contengono elementi di tipo auto. Le altre sono vuote; primaInCoda e ultimaInCoda sono gli indici delle celle che contengono rispettivamente i dati della prima e dell'ultima auto in coda.



Soluzione

```
typedef char stringa[8];
typedef enum {inDiagnosi, inManutOrd, inManutStraord, inUscita} stato;
typedef enum {tagliando, marmitta, cinghia, cilindri, freni} tipoIntervento;
typedef struct {
  stringa targa;
  stato s;
  tipoIntervento interventoCorrente;
  tipoIntervento interventiPrec[100];
} auto;

typedef struct {
  auto elenco[20];
  int primaInCoda; /* indica la posizione della prima auto in coda */
  int ultimaInCoda; /* indica la posizione dell'ultima auto in coda */
} listaAuto;
listaAuto lista;

/* frammento di codice che risolve il punto b. Implementa un meccanismo detto
"coda circolare" per mantenere l'ordine tra le auto e poter riutilizzare i posti
che si liberano man mano che le auto vengono analizzate */
if(lista.primaInCoda!= lista.ultimaInCoda)
/* se i due indici sono uguali vuol dire che non ci sono altre auto in coda */
{
```

```
  lista.primaInCoda=lista.primaInCoda-1;
  if(lista.primaInCoda < 0) lista.primaInCoda = 19;
}
```

Esercizio 4 (8 punti)

Le prestazioni della memoria cache di un server aziendale sono le seguenti:

- Hit Rate = 80%
- Tempo medio di accesso ai dati 90ns

Rispondere alle seguenti domande, giustificando le risposte:

a) Calcolare il Miss Penalty della memoria cache, sapendo che l'Hit Time è pari a 15ns
b) Consiglierestе l'acquisto di una nuova cache con Hit Time pari a 10ns e Miss Penalty 400ns?

Soluzione

a) Il tempo medio è calcolato come T = HR*HT+(1-HR)*MP, quindi:
90 = 0.8*15+(1-0.8)*MP => MP = (90-0.8*15)/0.2 => MP = 390ns

b) Si tratta di un buon acquisto perché il tempo medio di accesso diventa pari a:
T' = 0.8*10+(1-0.8)*400 = 8+80 = 88ns < 90ns