Politecnico di Milano—Facoltà di Ingegneria Industriale 19 febbraio 2004

Appello di Informatica B

Prof. Angelo Morzenti - Prof. Vincenzo Martena

Cognome e nome: _____
Matricola: _____
Tipo di prova: ☐ **recupero I prova** ☐ **recupero II prova**

Parte I				Parte II					
1	**2(a)**	**2(b)**	**3**	**1(a)**	**1(b)**	**2**	**3(a)**	**3(b)**	**Totale**

*Verrà corretto solo quanto consegnato su questi fogli: **NON** è consentito consegnare fogli addizionali.*
Si forniscano risposte il più possibile concise e pertinenti. Si usi una grafia facilmente leggibile.

Prima Parte

1. È stata introdotta una nuova tecnologia per la costruzione dei calcolatori, in cui il componente elementare dei registri e delle memorie non possiede due valori, bensì tre; i nuovi calcolatori non sono perciò basati sulla nozione di bit (con valore 0 o 1) bensì su quella di trit (con valore 0, 1 o 2).

   (a) Quale intervallo di numeri naturali (con inizio da 0) è possibile codificare con un tryte (una combinazione di 8 trit)?

   ```
   [0 ...  6560]
   ```

   (b) Quanti trit deve contenere il registro indirizzi di un calcolatore basato su questa nuova tecnologia per poter indirizzare 500.000 parole?

   ```
   12
   ```

2. (a) Si scrivano le strutture dati necessarie per rappresentare il tipo di dato PIANETA (caratterizzato da nome, circonferenza e numero di satelliti) e il tipo di dato STELLA contenente il nome della stella e un array contenente i pianeti orbitanti attorno alla stella (al massimo 15) e un intero che indichi i pianeti effettivamente presenti.

   (b) Si scriva un programma che utilizzando le strutture definite in precedenza e con riferimento a una variabile `stl` di tipo STELLA stampi il nome di ogni pianeta che abbia un numero di satelliti superiore alla somma dei satelliti dei pianeti che sono più vicini di esso alla stella. Per semplicità si supponga che i dati della stella e dei relativi pianeti siano stati precedentemente inizializzati e i pianeti siano ordinati in senso crescente rispetto alla distanza dalla stella.

Soluzione esercizio 2

```
#include <stdio.h>
#define MAX_STRING 20


typedef struct {
  char nome[MAX_STRING];
  float circonferenza;
  int numeroSatelliti;
} PIANETA;

typedef struct {
  char nome[MAX_STRING];
  PIANETA pianeti[15];
  int numPianeti;
} STELLA;

void main() {
  STELLA stl;
  int i;
  int contatore = 0;

  /* Suppongo che stl sia già stata inizializzata */
  for(i = 0; i < stl.numPianeti; i = i + 1) {
    if(stl.pianeti[i].numeroSatelliti > contatore) {
      printf("%s\n", stl.pianeti[i].nome);
    }

    contatore = contatore + stl.pianeti[i].numeroSatelliti;

  }

}
```

3. Si consideri il seguente programma.

```
#include <stdio.h>

void main() {

  int a, b;
  int *p, **pp;

  a = 5;
  p = &b;

  b = a - 1;
  pp = &p;

  printf("%d %d %d %d\n", a, b, *p, **pp);

  *p = **pp + 2;
  p = &a;

  printf("%d %d %d %d\n", a, b, *p, **pp);

}
```

   Si riporti l'output fornito dal programma.

Soluzione esercizio 3

L'output del programma è il seguente:

```
5 4 4 4
5 6 5 5
```

Seconda Parte

1. Si consideri la seguente funzione:

```
int f(int x) {

   if(x < 2) {
      return 1;
   }
   else if (x == 2) {
      return 2;
   }
   else if (x % 2 == 0) {
      return 2 * f(x / 2);
   }
   else {
      return f(x - 1);
   }

}
```

   (a) Si scrivano i valori restituiti dalla funzione in seguito alle seguenti invocazioni:
   - `f(35)` restituisce `32`
   - `f(30)` restituisce `16`
   - `f(32)` restituisce `32`

   (b) Considerando i valori ottenuti al punto precedente, si indichi l'operazione compiuta dalla funzione quando riceve in ingresso un numero intero positivo.

   La funzione restituisce la più grande potenza di due minore o uguale all'argomento passato.
   In termini più rigorosi:

$$f(n) = 2^{\lfloor \log_2 n \rfloor}$$

2. Considerando la seguente struttura dati:

```
typedef struct el {
   int elem;
   struct el* next;
} elem_lista;
```

si scriva una funzione che abbia un parametro d'ingresso di tipo puntatore a elem_lista, rappresentante il puntatore al primo elemento di una lista e che modifichi la relativa lista eliminando tutti gli elementi di valore inferiore al proprio predecessore. Ad esempio se la lista contiente i seguenti elementi (2, 6, 3, 9, 6) la lista risultante sarà (2, 6, 9). La funzione termina resituendo il puntatore al primo elemento della lista.

Soluzione esercizio 2

```
elem_lista* funz(elem_lista* lista) {
  elem_lista* corr, *prec;

  if(lista == NULL) {
    return NULL;
  }
  else {
    prec = lista;
    corr = lista->next;
    while(corr != NULL) {
```

```
      if(corr->elem < prec->elem) {
        prec->next = corr->next;
        free(corr);
        corr = prec->next;
      }
      else {
        prec = corr;
        corr = corr->next;
      }
    }
    return lista;
  }
}
```

3. Siano date le seguenti strutture dati per rappresentare il tipo di dato PARTITA contenente il nome della SquadraA e della SquadraB (entrambi array di 20 caratteri), il numero di goal della squadraA e il numero di goal della squadraB e il tipo di dato CLASSIFICA (contenente un array con i nomi 18 squadre, e un array di interi rappresentante i punti conseguiti dalla varie squadre)

```
typedef struct {
    char SquadraA[20];
    char SquadraB[20];
    int goalSqA, goalSqB;
} PARTITA;

typedef struct {
    char squadre[18][20];
    int punti[18];
} CLASSIFICA;
```

Utilizzando queste strutture dati definite si implementino le seguenti funzioni:

(a) `risultato` che riceve in ingresso una partita e restituisce 1 in caso di vittoria della SqaudraA, -1 in caso di vittoria della squadraB e 0 in caso di pareggio.

(b) `giornata` che riceve in ingresso un array di 9 partite e un parametro di tipo classifica contenenente i nomi delle squadre e i punti conseguiti nelle precedenti giornate di campionato. La funzione deve modificare opportunamente i punteggi delle squadre sulla base dei risultati delle partite (si ricordi che in caso di vittoria vengono assegnati 3 punti alla squadra vincente e 0 alla perdente mentre in caso di pareggio viene assegnato un punto ad entrambe le squadre). A tale scopo la funzione`giornata` deve far uso della funzione `risultato` e della funzione (assunta già implementata) `int nomiUguali(char s1[], char s2[])` che restituisce 1 se le due stringhe s1 e s2 coincidono, 0 altrimenti.

Soluzione esercizio 3

```
typedef struct {
  char SquadraA[20];
  char SquadraB[20];
  int goalSqA, goalSqB;
} PARTITA;

typedef struct {
  char squadre[18][20];
  int punti[18];
} CLASSIFICA;

int risultato(PARTITA p) {
  if(p.goalSqA > p.goalSqB) {
```

```
    return 1;
  }
  else if(p.goalSqA < p.goalSqB) {
    return -1;
  }
  else {
    return 0;
  }
}

/* funzione ausiliaria: restituisce la posizione occupata dalla
   squadra nell'array, -1 se la squadra non è stata trovata */
int cercaSquadra(CLASSIFICA c, char nomeSquadra[]) {
  int j;

  for(j = 0; j < 9; j = j + 1) {
    if(nomiUguali(c.squadre[j], nomeSquadra) == 1) {
      return j;
    }
  }
  return -1;
}


void giornata(PARTITA partite[], CLASSIFICA *c) {
  int i, j;
  int posSquadra;

  for(i = 0; i < 9; i = i + 1) {
    if(risultato(partite[i]) == 1) {
      posSquadra = cercaSquadra(*c, partite[i].SquadraA);
      if(posSquadra != -1) {
        c->punti[posSquadra] = c->punti[posSquadra] + 3;
      }
    }
    else if(risultato(partite[i]) == -1) {
      posSquadra = cercaSquadra(*c, partite[i].SquadraB);
      if(posSquadra != -1) {
        c->punti[posSquadra] = c->punti[posSquadra] + 3;
      }
    }
    else {
      posSquadra = cercaSquadra(*c, partite[i].SquadraA);
      if(posSquadra != -1) {
        c->punti[posSquadra] = c->punti[posSquadra] + 1;
      }
      posSquadra = cercaSquadra(*c, partite[i].SquadraB);
      if(posSquadra != -1) {
        c->punti[posSquadra] = c->punti[posSquadra] + 1;
      }

    }
  }
}
```