	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **FONDAMENTI DI INFORMATICA** Appello 6 Luglio 2016	COGNOME E NOME
		MATRICOLA
		Spazio riservato ai docenti

- Il presente plico contiene 4 esercizi e **deve essere debitamente compilato con cognome e nome, numero di matricola**.
- Il tempo a disposizione è di 1 ora e 30 minuti.
- 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 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 telefoni o pc**. Chi tenti di farlo vedrà annullata la sua prova.
- 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**.

Esercizio 1 (9 punti)

Si consideri una lista dinamica di interi definita come:

```
struct nodo
{
      int val;
      struct nodo *next;
};

typedef struct nodo *lista;
```

Implementare la seguente funzione:

```
void ruota(lista *l);
```

che riceve in ingresso la testa di una lista `l` di interi, la modifica spostando l'ultimo elemento della lista `l` in testa. Ad esempio, la lista 2->5->7->9 viene modificata in 9->2->5->7.

Note. Nel caso la lista `l` sia vuota, la sua testa contiene il valore NULL. Se necessario, è possibile implementare ulteriori funzioni di supporto da utilizzare all'interno della funzione `ruota`.

Soluzione

```
void ruota(lista *l)
{
      lista pre;
      lista cur;

      if (*l!=NULL && (*l)->next!=NULL)
      {
            pre = *l;
            cur = (*l)->next;
            while (cur->next!=NULL)
            {
                  pre = cur;
                  cur = cur->next;
            }

            pre->next = NULL;
            cur->next = *l;
            *l = cur;
      }
}
```

Esercizio 2 (9 punti)

Scrivere un programma C che legge due file di testo "in1.txt" e "in2.txt" contenenti una sequenza di valori float (un valore per riga) in ordine crescente (ogni file contiene al massimo 100 valori). Il programma dovrà quindi stampare a video tutti i valori contenuti in entrambi i file in modo che la sequenza stampata sia in ordine crescente.

Nota. Non è possibile ipotizzare che il numero di valori contenuti nei due file di ingresso "in1.txt" e "in2.txt" sia uguale. Si ipotizzi che l'apertura dei due file di testo vada sempre a buon fine. Se lo si ritiene necessario per lo svolgimento del programma, è possibile memorizzare il contenuto dei file all'interno di opportuni array.

Soluzione

```
#include <stdio.h>

 int main()
 {
      FILE *in1, *in2;
      float v1[100], v2[100];
      int n1=0, n2=0, i=0, j=0;

      in1 = fopen("in1.txt","r");
      in2 = fopen("in2.txt","r");

      while( !feof(in1) )
      {
            fscanf(in1,"%f",v1+n1);
            n1++;
      }

      while( !feof(in2) )
      {
            fscanf(in2,"%f",v2+n2);
            n2++;
      }

      while (i<n1 && j<n2)
      {
            if (v1[i]<v2[j])
            {
                  printf("%f\n",v1[i]);
                  i++;
            }
            else
            {
                  printf("%f\n",v2[j]);
                  j++;
            }
      }

      while (i<n1)
      {
            printf("%f\n",v1[i]);
            i++;
      }

      while (j<n2)
      {
            printf("%f\n",v2[j]);
            j++;
      }

      fclose(in1);
      fclose(in2);

      return 0;
 }
```

Esercizio 3 (5 punti)
Un sistema dispone di 4 Mbyte di memoria virtuale e una memoria fisica indirizzabile con paginazione, caratterizzata dai seguenti parametri: indirizzo fisico di 20 bit e 512 pagine di memoria con parole di memoria di 1 byte.

Rispondere alle seguenti domande giustificando le risposte:

A) Quale è la dimensione della memoria fisica indirizzabile?
B) Quale è la struttura dell'indirizzo virtuale e di quello fisico, e la lunghezza dei campi che li costituiscono?
C) Un consulente afferma che aumentando la dimensione delle pagine di memoria e quindi riducendo il numero di pagine di memoria, diminuiscono i possibili sprechi di memoria. Siete d'accordo con questa affermazione? Argomentare in maniera adeguata la propria risposta.

Soluzione

A) 20 bit -> 2^20 byte di memoria fisica -> 1 Mbyte

B)
512 (2^9) pagine fisiche -> 9 bit per NPF
4 Mbyte memoria virtuale -> 22 bit indirizzo virtuale

Quindi si ha,

NPV: 11 bit, offset: 11 bit
NPF: 9 bit, offset: 11 bit

C) L'affermazione è errata: aumentando la dimensione delle pagine di memoria, e quindi riducendo il numero di pagine di memoria, si ha meno controllo sull'utilizzo della memoria, dovendo assegnare blocchi contigui (una pagina) più grandi a un unico processo, aumenta la probabilità di sprecare spazi durante il funzionamento del sistema.

Esercizio 4 (9 punti)

A) Dichiarare un tipo di dato audiolibro che consenta di rappresentare le seguenti informazioni: titolo del libro, autore del libro, narratore del libro, durata (ore, minuti, secondi) e la lista dei capitoli che compongono l'audiolibro (al massimo 100). Per ciascun capitolo dell'audiolibro, è necessario memorizzare inizio (ore, minuti, secondi), fine (ore, minuti, secondi) e durata del capitolo (ore, minuti, secondi).

B) Scrivere una funzione C, `lunghi`, che riceve in ingresso una variabile di tipo audiolibro e restituisce il numero di capitoli in esso contenuto che hanno durata superiore a 30 minuti.

Soluzione

A)

```
typedef struct
{
      int ore;
      int minuti;
      int secondi;
} tempo;

typedef char stringa[100];

typedef struct
{
      tempo inizio;
      tempo fine;
      tempo durata;
} capitolo;

typedef struct
{
      stringa titolo;
      stringa autore;
      stringa narratore;
      tempo durata;
      capitolo capitoli[100];
      int ncapitoli;
} audiolibro;
```

B)

```
int lunghi(audiolibro a)
{
      int i,n;
      for (i=0; i<a.ncapitoli; i++)
            if (a.capitoli[i].durata.ora*60 + a.capitoli[i]. durata.minuti > 30)
                  n++;

      return n;
}
```