Fondamenti di Informatica - Ingegneria delle Telecomunicazioni – Prof. Maristella Matera

Appello – 3 Luglio 2015

Il tempo massimo a disposizione per svolgere la prova è di **2h e 15m**. Non è permessa la consultazione di alcun materiale didattico ed è vietato utilizzare calcolatrici, telefoni, PC. **Il voto minimo per superare la prova è 18**.

Esercizio 1 – Liste dinamiche (10 punti)

Sia data una lista dinamica che memorizza una sequenza di abbreviazioni (per esempio, "SI", "MI", "PC") e per ognuna di esse le corrispondenti parole estese (per esempio, "Sistema Informativo", "Milano", "Per Conoscenza"). Ogni nodo della lista è quindi una struttura definita nel seguente modo:

```
struct nodo {
    char sigla[2];
    char parola[20];
    struct nodo *next;
}
```

Si scriva una funzione in C che riceve come parametri **il puntatore alla testa della lista** (la lista non deve essere creata, si suppone che esista già in memoria), **una abbreviazione** e **la parola corrispondente** e ricerca l'abbreviazione nella lista; quindi:

1. Se la sigla non è presente nella lista, inserisce in coda alla lista un nuovo elemento per memorizzare la sigla e la parola estesa ricevute come parametro e restituisce 0; per esempio, se la lista contiene le coppie sopra citate (<"SI", "Sistema Informativo" >, <"MI", "Milano" >, <"PC", Per Conoscenza>) e la sigla da cercare è "CC", allora la funzione restituirà il valore 0.

2. Se la coppia <sigla, parola> è presente nella lista, la funzione restituisce 1; per esempio se la funzione riceve come parametri "SI" e "Sistema Informativo" allora restituisce il valore 1;

3. Se la sigla è contenuta nella lista ma è associata a una parola differente, la funzione restituisce 2; per esempio, data la lista precedente, se la funzione riceve come parametri "SI" e "Sicurezza Informatica", allora restituisce il valore 2.

Esercizio 2 – Funzioni in C e lettura da file (10 punti)

Si supponga di avere a disposizione un file che memorizza una sequenza (di lunghezza ignota) di valori interi. Nel file i diversi valori sono separati da uno spazio.

Si scriva una funzione in C che riceve come parametri **il nome del file** (una stringa) e due valori interi, **v1** e **v2** (con v1 minori v2), e modifichi il file nel modo seguente:

- i **valori minori di v1** devono essere memorizzati tutti **all'inizio del file**.
- i **valori maggiori o uguali a v1 e minori di v2** devono essere memorizzati tutti **al centro del file**.
- i **valori maggiori o uguali a v2** devono essere memorizzati **alla fine del file**.

L'ordine all'interno di ciascuna partizione del file deve rimanere lo stesso che nel file iniziale; cioè le partizioni non devono essere ordinate.

Per esempio, se il file iniziale contiene la sequenza:

2 5 21 8 10 1 4 16 3

e i valori di v1 e v2 sono rispettivamente 3 e 9, il file diventa

2 1 5 8 4 3 21 10 16

N.B.: è possibile far uso di strutture dati di appoggio per creare le tre partizioni prima di memorizzarle nel file.

Esercizio 3 – Funzioni ricorsive (8 punti)

1. (4 punti). Scrivere una funzione ricorsiva in C che, ricevuto come parametro un valore intero **n**, calcoli i numeri interi **T(n)** definiti dalle seguenti relazioni:

$T(0) = 0, T(1) = 1$

$T(n) = 2T(n-2) + 5 \quad$ per $n \geq 2$

2. (4 punti). Scrivere una funzione ricorsiva in C che, ricevendo come parametri un array **a** di interi positivi**,** la sua dimensione **n,** e un valore **val**, restituisca 1 se tutti gli elementi dell'array sono maggiori di val, 0 altrimenti.

Esercizio 4 – Processi (6 punti)

Si consideri il seguente codice, contenuto nel file **processi.c** :

```
#include <stdio.h>

int main(int argc, char **argv){
    int i=0, j=0;
    if (argc<2) return -1;
        for (i=0; i < 4; i=i+2){
        int pid=fork();
        if (!pid){
            printf("%c\n", getpid(),argv[1][i]);
            i=4;
        } else {
            wait();
        }
    }
    return i;
}
```

Si supponga che il programma sia invocato con la linea di comando
./a.out <matricola>
Si assuma inoltre che il processo bash dal quale il programma viene invocato abbia pid 500 e che i pid dei vari processi creati siano calcolati in modo progressivo.

Si illustri:
1. L'output del programma
2. La gerarchia dei processi (cioè quali processi figli vengono creati e chi li genera) che si viene a creare durante l'esecuzione del programma.