Laboratorio di Algoritmi e Strutture Dati I Docente: Murano

-- -- -- 18 DICembre 2008 -- -- --

Laurea in Informatica Università degli Studi di Napoli "Federico II"

Nome e Cognome Numero di Matricola:

Spazio riservato alla correzione

1	2	3	4.	Totale
/6	/6	/10	/12	/34

Non utilizzate altri fogli. Utilizzate soltanto lo spazio sottostante. Fogli differenti non saranno presi in considerazione per la correzione. Non scrivere a matita. Per tutti gli esercizi, descrivere la complessità asintotica delle funzioni implementate

1. Si consideri uno Stack **S**, implementato con array **S[MAX+1]**. Si implementi la funzione ricorsiva **moltiplica** che sostituisce ogni elemento di posizione **i** con il prodotto di tutti gli elementi che lo precedono nello stack (dal bottom all'elemento i). Si ricordi che lo stack è una struttura dati che permette l'accesso ai suoi dati solo dal top. Le funzioni di gestione stack possono essere omesse.

   Esempio: stack iniziale 4|3|1|2|5 (4 bottom dello stack) – stack finale 4|12|12|24|120|

2. Si considerino due liste di numeri interi **Lista1 e Lista2 "ordinate in modo strettamente crescente"** implementate come liste doppiamente puntate e non circolari, utilizzando la seguente struttura

```
struct elemento {
        struct elemento *prev;
        int inf;
        struct elemento *next;}

struct elemento *Lista1,*Lista2;
```

Si implementi una funzione (**possibilmente ricorsiva**) che rimuova da Lista1 tutti i numeri presenti in Lista2, senza usare strutture dati aggiuntive e senza cambiare l'ordine degli elementi. Restituire solo Lista1.

Esempio, sia **Lista1** uguale a 1→2→4→7→10 e **Lista2** uguale a 2→3→7→11. Dopo l'esecuzione di remove, **Lista1** sarà uguale a **1→4→10**.

3. Sia T un albero binario di ricerca, implementato con la seguente struttura a puntatori:

```
struct nodo {
        int inforadice;
        struct nodo *left;
        struct nodo *right;}

struct nodo *T;
```

a. Dato un albero P implementato con la stessa struttura di T, implementare una funzione in linguaggio C che verifichi, senza usare strutture dati di appoggio, che T e P sono identici (struttura e valori);
b. implementare una funzione in linguaggio C che verifichi (possibilmente senza strutture di appoggio), che per ogni elemento di T ci sia il suo predecessore o il suo successore.

4. Siano **G** e **H** due grafi orientati pesati entrambi con pesi positivi, di **n** vertici 0, 1,…, n-1 e rappresentati con liste di adiacenza utilizzando la seguente struttura:

```
typedef struct graph {
        int   nv;
        edge   **adj; } graph;

        graph *G, *H;
```

```
typedef struct edge  {
        int   key;
        int   peso;
        struct  edge   *next; } edge;
```

scrivere in linguaggio C una funzione che, presi in input i due grafi **G e H,** restituisca un nuovo grafo P (della stessa struttura di G e H), che abbia

   i. gli stessi vertici di G e H
   ii. un arco (a, b) per ogni arco arco (a,b) presente sia in H che nella trasposta di G, e tale che la differenza tra i due pesi sia positiva. Tale differenza rappresenta poi il peso di (a,b) in P.

b. Implementare una funzione che calcoli, dato G e un suo nodo, il suo grado incidente.