II esonero
PROGRAMMAZIONE II - 31/5/2006
Prof. E Fachini

Cognome: _____
Nome: _____
N. Matricola: _____

Esercizio n. 1 (sbarramento)

Si definisca una funzione C che dato in input un albero binario e un intero k, restituisce 1 se i nodi del livello k nell'albero costituiscono una sequenza crescente. Può servire l'uso di INT_MIN, il minimo intero rappresentabile. Si introducano pre e post condizioni.

Soluzione esercizio 1:

```
int livKOrd( TreePtr t, int k)
/*prec: k>=0,
postc: restituisce 1 se i nodi presenti al livello k sono ordinati in
ordine crescente o l'albero è vuoto,0 altrimenti*/
{int corr= INT_MIN;
if (!t) return 1;
return livkAus(t,k,&corr);
}
int livkAus(TreePtr t, int k, int* corr);
int livkAus(TreePtr t, int k, int* corr)
{if (!t) return 1;
if (k==0) {if (*corr < t->elem) {*corr = t->elem; return 1;} else
return 0;}
return livkAus(t-> left,k-1,corr ) && livkAus(t->right,k-1,corr);}
```

II esonero
PROGRAMMAZIONE II - 31/5/2006
Prof. E Fachini

Cognome: _____
Nome: _____
N. Matricola: _____

Esercizio n. 2

Si producano dati di test, distinguendo tra quelli individuati seguendo l'approccio a scatola nera da quelli ottenuti a scatola trasparente, per la seguente funzione:

```
void cancLista(ListaPtr * L,int i);
/*prec: L != NULL
postc:cancella dalla lista gli elementi maggiori di i*/
```

A scatola nera:

Per ogni lunghezza n di lista, l'insieme delle liste di lunghezza n si divide tra quelle che restano invariate e quelle che vengono modificate.

Casi limite:
La lista vuota, per la quale il risultato è la lista vuota,
La lista con un solo elemento maggiore di i o minore di i.
Nel seguito assumiamo che i dati eliminati sono maggiori di i.
Casi da considerare:
Per liste di lunghezza due otteniamo 4 casi:
I due elementi vengono eliminati
Un solo elemento viene eliminato: il primo o il secondo
Nessun elemento viene eliminato.
Per liste di lunghezza tre otteniamo 8 casi:
I tre elementi vengono eliminati
Un solo elemento viene eliminato: il primo, il secondo o il terzo
Due elementi vengono eliminati: i primi due, gli ultimi due e il primo e l'ultimo.
Nessun elemento viene eliminato.
Per liste di lunghezza quattro consideriamo i seguenti casi
I quattro elementi vengono eliminati
Un solo elemento viene eliminato, in tutte le possibili posizioni,
Due elementi vengono eliminati di nuovo in tutti i casi possibili
Tre elementi vengono eliminati sempre in tutti i casi possibili.
Nessun elemento viene eliminato.

A scatola trasparente:

```
void cancLista(ListaPtr* L,int i)
/*prec: L != NULL
postc:cancella dalla lista gli elementi maggiori di i*/
{ListaPtr temp;
if(!(*L)) return;
if ((*L)->next == NULL )
{if ( (*L)->elem > i) *L = NULL;
return;}
if (((*L) -> next) -> elem > i)
{temp = (*L) -> next;
printf("l'elemento da cancellare è %d\n", temp->elem);
(*L) -> next = (*L) ->next -> next;
free(temp);}
cancLista(&(*L)->next,i);}
```

Poichè i casi di uscita dei due if iniziali così come quelli che comportano una o due chiamate ricorsive con i diversi cammini computazionali, sono trattati nella scelta dei dati a scatola nera non aggiungiamo altri controlli, salvo aggiungere casi in cui le uscite per falso dalle istruzioni condizionali è dato da un elemento uguale a i.
Già su una lista di due elementi , entrambi da cancellare , la funzione si

comporta male. In effetti il primo elemento in una lista di lunghezza due non viene controllato e questo errore si potrebbe rilevare anche dall'esame del codice.