Scritto di Programmazione II, Canale A-D
Prof.ssa E. Fachini, Dott.ssa Chiara Petrioli
appello di settembre 2006
parte I

Nome **Cognome** **Num. Matricola**

Esercizio 1. (Sbarramento)
Si scriva una funzione C ricorsiva che, preso in input un numero intero, ne stampa le cifre decimali in ordine inverso, ovvero dalla meno significativa alla più significativa. La vostra soluzione è *tail recursive*? Si motivi la riposta.
Esempio. Se il numero intero ricevuto come parametro di input è 1234, la funzione deve stampare 4321.

Soluzione.

```
void invertiCifre (long n)
  /*postc: stampa a video le cifre di n  in ordine inverso. */
{
  printf("%d", n % 10);
  if (n / 10 != 0)
    invertiCifre(n / 10);
}
```

Esercizio 2.

Si definisca una funzione iterativa che visiti un albero binario in ordine simmetrico (inorder), utilizzando una pila. Le funzioni di gestione della pila vanno solamente ma completamente specificate, fornendo il prototipo e le eventuali pre e post condizioni.

```
/*In questa soluzione la pila P è dichiarata come variabile globale e quindi
inizializzata e distrutta  al di fuori della funzione*/

void TreeInorderVisit(TreePtr t)
/*  postc: visita inorder iterativa con l'ausilio di una pila P */
{TreePtr temp;
while (t)
    {push(t,P);
    t = t->left;}
while (!vuota(P))
    {temp = pop(P);
    visit(temp);
    temp = temp->right;
    while (temp)
        {push(temp,P);
        temp = temp->left;}
    }
}


int   vuota(const PilaP p);
/* da' vero se la pila  p e' vuota, falso altrimenti
*prec: p é una pila valida
```

```
postc: restituisce un valore !=0  se la pila è vuota, 0 altrimenti*/

void      push(int el, PilaP p);
/* inserisce el in cima alla pila  p
prec: p è una pila valida e non piena
postc: el è in cima alla pila, se c'è abbastanza memoria, esce dal programma
altrimenti*/

int     pop(PilaP p);
/* elimina l'elemento in cima a p, se non e' vuota
*prec: p è una pila valida e non vuota
post: restituisce, eliminandolo, l'elemento in cima alla pila*/
```

Esercizio 3.

Si scriva una funzione che, mediante scambi della posizione dei figli dei nodi di un albero binario (sottoalbero destro che diventa sottoalbero sinistro e viceversa), riorganizza un albero in modo tale che, per ogni nodo, il sottoalbero sinistro contiene un numero di nodi maggiore o uguale di quello dei nodi del sottoalbero destro. Analizzare la complessità computazionale della soluzione proposta. *Nota.* E' possibile fornire una soluzione che risolve l'esercizio in tempo lineare.

Soluzione

```
int riordina(TreePtr t)
/*postc: restituisce il numero dei nodi di t e lo riorganizza in modo tale che,
per ogni nodo,
*il sottoalbero sinistro contiene un numero di nodi maggiore o uguale di quello
dei nodi del sottoalbero destro.*/
{TreePtr temp;
int s,d;
if (!t) return 0;
s = riordina(t->left);
d = riordina (t->right);
if (s < d) /*scambio dei sottoalberi*/
{temp = t ->left;
t -> left  = t -> right;
t -> right = temp;}
return s+d+1;}
```

Esercizio 4

Si individuino opportuni dati di test per la seguente funzione, evidenziando per quali dati si e' seguito l'approccio a scatola nera e per quali quello a scatola trasparente. Si dia una breve motivazione della scelta.

```
/*individua la prima occorrenza del carattere c nella stringa s*/
char * Rstringchr (char *s, int c)
/*Pre: s!= NULL*/
/* Post: restituisce il puntatore alla prima occorrenza di c nella stringa. Se
tale occorrenza non esiste, restituisce NULL*/
```

```
{
    if (*s =='\0')
        return (NULL);
    else if (*s == c)
        return s;
    else
        return (Rstringchr(s+1,c));
}
```

Scelta dei dati a scatola nera

Dalle postcondizioni si hanno due casi:
1. la funzione restituisce il puntatore NULL quando il carattere non occorre come per esempio con s = “stampa” e c = ‘r’, come caso limite prendiamo una stringa di lunghezza 1 diverso dal carattere cercato oppure quando la stringa non ha caratteri: s=”\0” e c = ‘a’
2. la funzione restituisce un puntatore non NULL quando il carattere occorre nella stringa:
s = “stampa” e c = ‘m’, come caso limite prendiamo una stringa di lunghezza 1 che contiene il carattere cercato. Per esempio s= “a” e c=’a’.

A scatola trasparente:

La condizione di uscita dal primo if è già soddisfatta dalla stringa vuota scelta precedentemente, così come anche l’uscita dal primo if senza chiamate (una stringa di lunghezza 1 che contiene il carattere cercato).
Verifichiamo se con una stringa di lunghezza almeno due troviamo c nelle due posizioni possibili, per esempio con s= “ai” e c =’a’ e poi c=’i’.
Nel primo caso non ci sono altre chiamate e nel secondo una sola .
Passiamo a stringhe di lunghezza 3 e prevediamo una scelta di c per ogni posizione, per esempio: s= “ape” e c=’a’, c=’p’ e c = ‘e’. Qui nel secondo e terzo caso abbiamo una rispettivamente due chiamate.
Infine testiamo su una o più stringhe “lunghe”, sempre tutti i possibili casi di c, Per esempio
S = “qwertyui12345plkjhòàè+’’ì<.,xbvm”