Programmazione II canale AD -- Esonero del 21/04/2006

Cognome **Nome** **N. Matricola**

……

Esercizio 1 (Sbarramento)

Si scriva una funzione ricorsiva che riceve in input una lista concatenata di interi L e restituisce una nuova lista concatenata di interi L1 in cui l'elemento j-simo (j ≥1) contiene la somma degli elementi della sottolista di L ottenuta escludendo i primi j-1.

Esempio:

Input: 20 -> 4 -> 5 -> 10 -> 9
Output: 58 -> 28 -> 24 - > 19 - > 9

Soluzione

```
typedef struct listElem *list;
typedef struct listElem {
    int     info;
    list    next;
} listElem;

list insInTesta(int n, list ls)
/*postc: l'elemento contenente n è aggiunto alla lista ls */
    {list newls = malloc(sizeof(listElem));
    assert(newls != NULL);
   /* inserimento in testa */
newls->info = n;
newls->next = ls;
return newls;
}

list sommaCoda(list ls)
{ /* restituisce una lista in cui ogni elemento contiene la somma degli elementi
della lista in input nelle posizioni ad esso successive */
list newls = NULL;
int s;
if (ls != NULL)
    {newls = sommaCoda(ls->next);
        /* se newls non NULL, la testa e' la somma della coda
         * altrimenti la somma della coda e' 0*/
         s = newls != NULL ? newls->info : 0;
        /* inserisco nuovo elemento */
        newls = insInTesta(ls->info + s, newls);
    }
return newls;
}
```

Programmazione II canale AD -- Esonero del 21/04/2006

Cognome **Nome** **N. Matricola**

……

Esercizio 2 (Code e Stack)

Si supponga di avere due stack A e B di interi, implementati con liste concatenate semplici dalle seguenti dichiarazioni:

```
typedef struct elem{
      int               data;
      struct elem       *next;
      } elem;

typedef struct stack {
      int   cnt;
      elem  *top;
} stack;

stack A, B;
```

Si definiscano, specificando le ipotesi necessarie, precondizione e postcondizione, tre funzioni che simulino le operazioni di accoda (**enqueue),** estrai **(dequeue) e (**primo**) front** in una coda di interi usando i due stack A, e B. In particolare lo stack A deve essere usato per contenere i dati della lista e il B come stack ausiliare per le operazioni necessarie alle simulazione. La coda è definita dalla seguente dichiarazione:

```
typedef struct coda{
            int   cnt;        /* # elementi attuali nella coda */
            elem  *testa;     /* punt. all' elemento in  testa alla coda */
            elem  *coda;      /* punt all' elemento in coda alla coda */
} coda;
```

Soluzione.

```
/*IPOTESI
-------
Considereremo uno coda memorizzata in uno stack in modo tale che
l'elemento nella coda corrisponda all'elemento nel bottom dello stack
e l'elemento  in testa alla coda a quello nel top dello stack.


ESEMPIO
-------
CODA: NULL<---1<---2<---3<---4<---5
        [coda]                [testa]
STACK:  5--->4--->3--->2-->1--->NULL
      [top]             [bottom]


FUNZIONI AUSILIARI SU STACK
---------------------------
Si considerano date le seguenti funzioni  su
```

```
stack, di cui riportiamo solo i prototipi: */
typedef stack* PilaP; /*è il tipo del puntatore a una struttura per gli elementi
della pila*/

int   vuota(const PilaP p);
/* da' vero se la pila  p e' vuota, falso altrimenti
*prec: p è una pila creata con costrPila
postc: restituisce un valore !=0  se la pila è vuota, 0 altrimenti*/


int   piena(const PilaP p);
/* da' vero se la  pila p e' piena, falso altrimenti
*prec: p è una pila creata con costrPila
postc: restituisce un valore !=0  se la pila è piena, 0 altrimenti*/


void      push(int el, PilaP p);
/* inserisce el in cima alla pila  p
prec: p è una pila creata con costrPila 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 creata con costrPila e non vuota
post: restituisce, eliminandolo, l'elemento in cima alla pila*/

int      top(const PilaP p);
/* legge e restituisce l'elemento in cima a p, se non e' vuota
*prec: p è una pila creata con costrPila e non vuota
post: restituisce l'elemento in cima alla pila*/



/*Inoltre consideriamo la seguente funzione che travasa
uno stack sorgente in uno stack destinazione;*/

void travasa(PilaP source, PilaP dest)
/*prec: source e dest sono  pile create con costrPila
postc: dest contiene gli elementi di source in ordine inverso  */
{
    int d;
    while (!vuota(source))
    {d = pop(source);
        if (!piena(dest))
            push(d,dest);
    }
}


/*ENQUEUE, DEQUEUE, FRONT
Il tipo CodaP è il tipo puntatore per la coda.
Date le ipotesi e considerando le due variabili di tipo stack A e B come
```

```
globali, le operazioni possono realizzarsi nel seguente modo:*/

typedef coda* CodaP;

void enqueue( CodaP queue, int d){
/* Pre: queue è stata inizializzata e non è piena  */
/* Post:  l'elemento d è inserito in coda alla coda*/
travasa(&A,&B);         /* copio il contenuto di A in B */
    push(d,&A);         /* inserisco l'elemento d in A */
    queue->coda = A.top;    /* memorizzo il suo indirizzo come coda della lista
*/
    travasa(&B,&A);         /* travaso il contenuto di B nel nuovo A */
    if (queue->cnt == 0);   /* solo se è il primo inserimento ...*/
        queue->testa = queue->coda; /* ... testa e coda sono l'unico elemento
della coda */
    queue->cnt++;       /* incremento il contatore di elementi nella coda */
}

int dequeue( CodaP queue){
/* Pre: queue è stata inizializzata e non è vuota */
/* Post: restituisce il primo della coda estraendolo */
    int d;
    d = pop(&A);        /* prelevo l'elemento dal top dello stack */
    queue->testa = A.top;   /* aggiorno il puntatore alla testa della coda */
    queue->cnt--;       /* aggiorno il contatore di elementi */
    return(d);      /* restituisco il dato */
}
int front( CodaP queue){
/* Pre: queue è non vuoto */
/* Post: restituisce il primo elemento della coda, se non vuota */
return(top(&A));  /* restituisco il dato nel top di A, quindi nella testa
                della coda, senza modificarla */
}
```

Programmazione II canale AD -- Esonero del 21/04/2006

Cognome **Nome** **N. Matricola**

……

Esercizio 3 (Tail Recursion)

Si scriva una funzione ricorsiva in coda che riceve in input una lista concatenata di stringhe e restituisce una lista di stringhe di circa metà elementi in cui l'i-simo (i≤1) elemento contiene la concatenazione degli elementi in posizione 2i-1 e 2i (se esiste) della lista originaria, che deve rimanere invariata.

Esempio:

input : ab -> cd -> ef
Output : abcd -> ef

Soluzione.

```
char* conc(char * s, char * t);
/*prec: s != NULL && t != NULL
* postc: restituisce, allocando la memoria, la concatenazione di s e t*/
char* conc(char * s, char * t)
{char* x;
x = (char*) malloc((strlen(s) + strlen(t)+1)*sizeof(char));
x = strcpy(x,s);
x = strcat(x,t);
return x;}

 ListeStrP divMetaConc(ListeStrP lista)
/* post: restituisce una lista contenente nell'elemento i-simo la stringa
ottenuta concatenando le stringhe
*contenute negli elementi di posto 2i-1 e 2i della lista data, lasciandola
inalterata*/
{ListeStrP app;
char* temp;
if (!lista || !(lista->nextPtr)) return lista;
app = (ListeStrP)malloc(sizeof(ListeStr));
assert(app);
app -> elem = conc(lista -> elem,lista->nextPtr -> elem);
app->nextPtr = divMetaConc(lista ->nextPtr ->nextPtr);
return app;}
```

Programmazione II canale AD -- Esonero del 21/04/2006

Cognome **Nome** **N. Matricola**

……

Esercizio 4 (Facoltativo. Ricorsione e Problem Solving)
Si progetti un algoritmo ricorsivo, basato sul divide et impera, che calcoli il minimo e il massimo in un vettore di n interi. Si codifichi l'algoritmo in C.

Soluzione.

```
MinMass* MinMax (int* vett, int i, int j)
/* prec i<=j;
postc: restituisce un puntatore a una struttura i cui campi interi contengono il
minimo e il massimo
* sugli elementi di vett tra vett[i] e vett[j], compresi*/
{MinMass *temp,*Mml,*Mmr;
int m;
temp =(MinMass*)malloc(sizeof(MinMass));
assert(temp);
if(i == j) {temp -> min = vett[i];temp -> max = vett[i];return temp;}
m = (j+i)/2;
Mml  = MinMax(vett,i,m);
Mmr  = MinMax(vett,m+1,j);
temp -> max = max( Mml -> max, Mmr -> max);
temp -> min = min( Mml -> min, Mmr -> min);
return temp;}
```