Fondamenti di Informatica T-1 (A.A. 2014/2015) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Giovedì 29 Gennaio 2015 – durata 1h
Totale 12 punti, sufficienza con 7

Compito B

ESERCIZIO 1 (6 punti)

Sono date due liste di interi `l1` e `l2`. La lista `l1` è una lista ordinata di interi non negativi mentre `l2` è una lista contenente interi positivi. Si realizzi la funzione ITERATIVA

```
list cerca(list l1, list l2);
```

che estragga gli elementi della lista `l2` nelle posizioni indicate dagli elementi della lista `l1` (il valore 0 indica il primo elemento, il valore 1 indica il secondo elemento...). Qualora `l1` contenga indici per posizioni non esistenti di `l2`, dovrà essere inserito il valore -1. Per esempio, se `l1= {1,4,10,12}`, `l2= {4,42,11,29,69,17}`, la funzione `cerca()` deve restituire la lista `l3= {42,69,-1,-1}`.

La funzione `cerca()` dovrà essere implementata utilizzando le primitive dell'ADT lista. Si realizzi inoltre una semplice funzione `main()` di prova che invochi correttamente la funzione `cerca()` creata.

ESERCIZIO 2 (2 punti)

Si consideri la seguente funzione

```
double f(int a){
      if (a==0){
            return 0;
      }else
            if(a==1){
                  return 1;
            }else{
                  return (f(a-2.0) + f(a-1.0));
            }
}
```

Mostrare la sequenza dei record di attivazione ed il valore di ritorno nel caso in cui la funzione sia invocata con parametri attuali `(3)`.

ESERCIZIO 3 (3 punti)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data).
Nota: si ricorda che l'operazione del modulo è il resto della divisione tra interi.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* conv(int n,int *out){

	int j=4;
	char *b;

	b = (char*)malloc((j+1)*sizeof(char));
	b[j]='\0';

	for(--j;n!=1;--j){
		if(j==0){
			*out=1;
			return b;
		}else{
			*(b+j)='0'+(n%2);
			n=n/2;
		}
	}
	b[j--]='1';
	for(;j>=0;--j){
		*(b+j)='0';
	}
	*out=0;
	return b;
}

int main(){

	int b=7;
	int o;
	char *r;

	r= conv(b,&o);

	printf("%d\n",o);
	if(o==0){
		printf("%s\n",r);
	}

	return 0;
}
```

ESERCIZIO 4 (1 punto)

Si descriva il funzionamento dell'algoritmo di ordinamento Merge Sort.

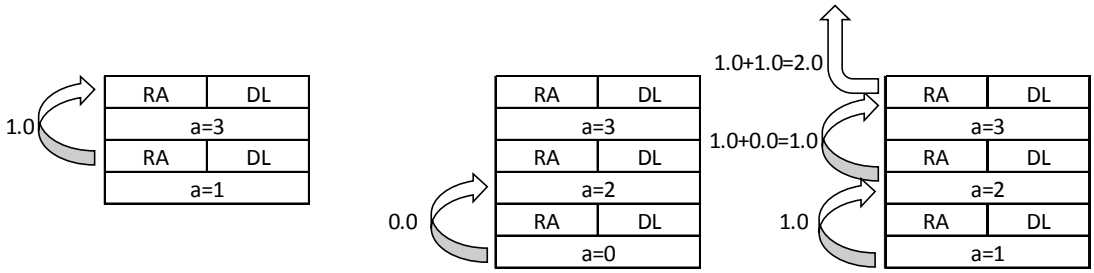
Soluzioni

ESERCIZIO 1

```
list cerca(list l1,list l2){

	int i=0;
	list temp = emptyList();
	list res = emptyList();

	while(!empty(l1)){
		if(empty(l2)){
			temp=cons(-1,temp);
			l1=tail(l1);
		}else{
			if(head(l1)==i){
				temp=cons(head(l2),temp);
				l1=tail(l1);
			}
			l2=tail(l2);
			i++;
		}
	}
	// la lista temp viene costruita “al contrario”, ora la ribaltiamo per
avere gli elementi della lista risultato nelle posizioni corrispondenti a quelle
degli elementi di l1.
	while(!empty(temp)){
		res=cons(head(temp),res);
		temp=tail(temp);
	}
	return res;
}

int main(){

	list valori,indici,risultato;

	valori=cons(4,cons(42,cons(11,cons(29,cons(69,cons(17,emptyList()))))));
	indici=cons(1,cons(4,cons(10,cons(12,emptyList()))));

	risultato = cerca(indici,valori);

	while(!empty(risultato)){
		printf("%d\n",head(risultato));
		risultato=tail(risultato);
	}

	return 0;
}
```

ESERCIZIO 2

La funzione restituisce il valore 2.



ESERCIZIO 3

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:

```
0
0111
```

La funzione `main()` inizializza una variabile intera `b` con valore `7` e dichiara due variabili: un intero `o` e un puntatore a carattere `r`. Invoca poi la funzione `conv()`,passando come argomento le variabili `b` e il riferimento a `o` e assegnando il risultato alla variabile `r`.

La funzione `conv()` inizializza una variabile intera `j` al valore `4` e dichiara un puntatore a carattere `b`. Questo viene fatto puntare ad un'area di memoria allocata dinamicamente dimensionata per contenere `j+1=5` valori `char` e assegna immediatamente il terminatore di stringa `'\0'` alla posizione `b[j]`, ovvero l'ultima. Esegue un ciclo `for` usando la variabile `j` come indice e iterando partendo dal valore ottenuto grazie all'operazione di pre-decremento (`3`) decrementandolo di uno fino a che l'argomento `n` della funzione è diverso da `1`. Ad ogni passo verifica che la variabile `j` sia uguale a `0`. In caso positivo, assegna alla variabile referenziata dal puntatore a `int out`, argomento della funzione, il valore `1` e ritorna il puntatore a carattere `b`. Altrimenti, assegna alla posizione `j`-esima della stringa `b` il carattere risultante dalla somma tra il carattere `'0'` e quello individuato dal risultato dell'operazione modulo tra `n` e il valore `2`. Successivamente, assegna a `n` il risultato della divisione intera tra il suo attuale valore e il numero `2`.
Terminato il ciclo, assegna alla posizione `j`-esima della stringa `b` il carattere `'1'`. L'accesso alla cella `j`-esima avviene con un'operazione di post-decremento sulla variabile `j`, aggiornandone il valore dopo che è stata usata per l'indirizzamento.
Si ha quindi un secondo ciclo `for`, privo di istruzione di inizializzazione, che continua a decrementare di `1` il valore di `j` ad ogni passo e si ripete finché la variabile `j` è pari o superiore a `0`. In questo ciclo, ad ogni passo viene inserito nella posizione `j`-esima della stringa `b` il carattere `'0'`. Terminato, assegna alla variabile referenziata dal puntatore a `int out`, argomento della funzione, il valore `0` e ritorna il puntatore a carattere `b`.

La funzione `main()` stampa sullo standard output il valore contenuto dalla variabile `o` (`0`) e, se `o` vale `0`, la stringa indirizzata dalla variabile `r` (`0111`).