Fondamenti di Informatica L-A (A.A. 2006/2007) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame di Mercoledì 13 Dicembre 2006 – durata 2h
Compito A

ESERCIZIO 1 (9 punti)

Si consideri un vettore (di dimensione nota) contenente dei valori interi maggiori di 0 ripetuti più volte. Tale vettore ha anche la caratteristica che, detto **`max`** il valore massimo presente, contiene almeno una volta ogni valore tra 1 e **`max`** compresi. Il candidato realizzi una funzione:

```
int * clean(int * v, int dim, int * newDim)
```

che, ricevuti come parametri il vettore **`v`** e la dimensione **`dim`** di tale vettore, restituisca come risultato un puntatore a **`int`** che rappresenta un nuovo vettore allocato dinamicamente. Tale vettore deve contenere gli elementi di **`v`** senza ripetizioni, esattamente nell'ordine con cui compaiono per la prima volta nel vettore **`v`**. Il candidato abbia cura di allocare dinamicamente solo la memoria strettamente necessaria. Tramite il parametro **`newDim`**, la funzione deve anche restituire la dimensione del vettore risultato.

Ad esempio, se **`v=[3,6,4,5,1,2,3,5]`**, il risultato deve essere un vettore di dimensione 6, della forma **`[3,6,4,5,1,2]`**.

ESERCIZIO 2 (9 punti)

Si supponga di avere a disposizione, già definiti, l'ADT lista per interi (denominato come al solito **`list`**, con relative primitive) e un nuovo ADT, denominato **`list_l`**, che rappresenta il concetto usuale di lista, definita però per elementi che a loro volta sono liste di interi. Il candidato supponga di possedere le usuali primitive (denominate però **`empty_l(…)`**, **`emptylist_l(…)`**, **`cons_l(…)`**, **`head_l(…)`**, **`tail_l(…)`**) definite opportunamente per il nuovo ADT **`list_l`**. Il candidato definisca una funzione

```
list flat(list_l ll)
```

che, ricevuta in ingresso una lista **`ll`** i cui elementi sono a loro volta liste di interi, restituisca in uscita una nuova lista di interi contenente tutti gli interi presenti nelle sotto-liste (in un qualunque ordine). Ad esempio, se invocata con parametro **`ll= [ [4,2,3,1], [5,6,7,12] ]`**, la funzione deve restituire come risultato la lista **`[4,2,3,1,5,6,7,12]`**. A tal scopo si utilizzino solo le operazioni primitive degli ADT **`list`** e **`list_l`**.

ESERCIZIO 3 (6 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)?

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

typedef struct item {
      char value;
      struct item * next;
} Item;
typedef Item * list;

int dim = 5;

list rotate(list l1) {
      list temp = l1;
      while (temp->next != NULL)
            temp = temp->next;
      temp->next = (list) malloc(sizeof(Item));
      temp->next->next = NULL;
      temp->next->value = l1->value;
      return l1->next; }

int main(void) {
      list l1, temp;
      int i;

      l1 = (list) malloc(sizeof(Item));
      l1->next = NULL;
      l1->value = 'e';
      for (i=1; i<dim; i++) {
            temp = (list) malloc(sizeof(Item));
            temp->next = l1;
            temp->value = 'e' - i;
            l1 = temp;
      }
      l1 = rotate(l1);
      while (l1 != NULL) {
            printf("%c ", l1->value);
            l1 = l1->next; }
      return (0); }
```

ESERCIZIO 4 (5 punti)

Si scriva una funzione ricorsiva **`int fun(char *str, char old, char new)`** che, ricevuti come parametri una stringa ben formata **`str`** e due char **`old`** e **`new`**, restituisca come valore di ritorno il numero di occorrenze del carattere **`old`** in **`str`** e sostituisca nella stringa **`str`** ogni occorrenza del carattere **`old`** con il carattere **`new`**.
Si proponga anche il codice di una possibile funzione che invochi **`fun()`**.

ESERCIZIO 5 (3 punti)
Si consideri la rappresentazione collegata usata per l'implementazione dell'ADT lista vista durante il corso. Si descrivano brevemente vantaggi e svantaggi delle scelte 1) di poter avere parziale condivisione di nodi (structure sharing) e 2) di non de-allocare mai nodi tramite **`free()`**.

ESERCIZIO 1

```
int * clean(int * v, int dim, int * newDim)
{
      int * result;
      int i, j, pos=0, trovato=0;

      *newDim = 0;
      for (i=0; i<dim; i++)
            if (v[i] > *newDim)
                  *newDim = v[i];
      result = (int*) malloc(sizeof(int)* *newDim);

      for (i=0; i<dim; i++)
      {     for (j=0;j<pos; j++)
                  if (v[i] == result[j]) trovato++;
            if (!trovato)
            {     result[pos] = v[i];
                  pos++;
            }
            trovato = 0;
      }
      return result;
}
```

ESERCIZIO 2

```
list flat(list_l l1)
{
      list result = emptylist();
      list temp;
      while(!empty_l(l1))
      {
            temp = head_l(l1);
            while (! empty(temp))
            {
                  result = cons(head(temp), result);
                  temp = tail(temp);
            }
            l1 = tail_l(l1);
      }
      return result;
}
```

ESERCIZIO 3

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

```
b c d e a
```

Nella funzione main(…) viene creata una lista di caratteri, contenenti nell'ordine gli elementi a,b,c,d,e. Viene invocata la funzione rotate(…) e la lista risultante viene poi stampata a video.

La funzione rotate(…) alloca memoria per un nuovo elemento e lo concatena in fondo alla lista. Quindi in tale ultimo elemento viene copiato il valore presente nel primo elemento della lista. La funzione restituisce come lista risultato il puntatore al secondo elemento.

ESERCIZIO 4

```
int fun(char* str, char old, char new){
      if(str[0]=='\0') return 0;
      else{
            if(str[0]==old){
                  str[0]=new;
                  return 1 + fun(&str[1],old,new);
            }
            else return fun(&str[1],old,new);
      }
}

int main(){
      int volte=0;
      char stringa[]="abcdefghelmeo";
      printf("Prima %s\n",stringa);
      volte=fun(stringa,'e','z');
      printf("Dopo: %d caratteri contati e nuova stringa= %s\n",volte,stringa);
      return 0;
}
```