Fondamenti di Informatica L-A (A.A. 2006/2007) - Ingegneria Informatica
Prof.ssa Mello & Prof. Bellavista – Prova d'Esame del 15/01/2007 - durata 2h

COMPITO B

ESERCIZIO 1 (11 punti)

Si consideri una lista di stringhe ben formate `stringhe` e un vettore di interi `ripetuti` avente spazio sufficiente a contenere un numero di interi uguale al numero di stringhe nella lista. Il candidato realizzi una funzione:

```
list caratteri(list stringhe, int *ripetuti, int* dim);
```

che restituisca una nuova lista con tutti e soli gli elementi della lista `stringhe` NON aventi caratteri ripetuti (ciascun carattere può essere presente in ogni stringa al più due volte). Inoltre la funzione deve inserire nel vettore `ripetuti` un intero per ciascuna stringa non inserita nella nuova lista; tale intero deve rappresentare il numero di caratteri ripetuti nella corrispondente stringa non inserita. Infine la funzione deve restituire tramite `dim` la dimensione logica del vettore `ripetuti`.
Ad esempio, se `stringhe = [andare, 1234, uk, sottomarini, studente]` la lista risultato sarà `[1234, uk]` (l'ordine degli elementi nella lista restituita è ininfluente) e il vettore `ripetuti = [1, 3, 2]`, in quanto `andare` ha un carattere ripetuto, `sottomarini` tre e `studente` due.

L'esercizio deve essere svolto utilizzando il tipo di dato astratto `list`, definito per le stringhe. Si possono utilizzare le sole operazioni primitive definite durante il corso, che quindi possono NON essere riportate nella soluzione. Non si possono usare altre funzioni di alto livello.

ESERCIZIO 2 (9 punti)

Usando le operazioni primitive dell'ADT stack di interi definite durante il corso (che NON devono essere riportate), realizzare una funzione ricorsiva:

```
void split(stack *x, stack *y);
```

che, dati due stack `x` e `y`, sposti in `x` (stack vuoto) tutti i dati presenti in `y` e minori di 0. L'ordine degli elementi di `x` e `y` deve rimanere necessariamente inalterato. Ad esempio, se `y = [-4, -5, 1, 5, -2, 2, 0, -3]`, dopo l'invocazione della funzione `split()`, `y` deve risultare `[1, 5, 2, 0]` e `x` deve valere `[-4, -5, -2, -3]`.

ESERCIZIO 3 (7 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>

int cont=8;

int add(int *int_v, char *char_v, char canc[]){
      int i,cont,tot;
      cont=0; tot=0;
      while(*(char_v)!='\0'){
            *(char_v-cont)=*char_v+(*int_v);
            char_v++;
            int_v++;
            tot++;
            for(i=0;i>=0 && *(canc+i)!='\0';i++){
                  if(canc[i]==*(char_v-cont-1)){
                        i=-2;
                        cont++;
                  }
            }
      }
      char_v[-cont]='\0';
      return tot-cont;
}

int main(){
      char cars[]="prodotti";
      int ints[]={1,3,-2,-2,0,1,2,-1};
      int i;
      i=add(ints,cars,"mnopq");
      printf("%d %s\n",i,cars);
      return 0; }
```

ESERCIZIO 4 (3 punti)
Si consideri la grammatica G con scopo S e simboli terminali {1,2,3,4,5,6,7}

```
S::= XMN | YNM
M::= XM | X
N::= YN | YMN | Y
X::= 5 | 6 | 7
Y::= 1 | 2 | 3 | 4
```

La stringa "3255127" appartiene alla grammatica? Se sì se ne mostri la derivazione left-most.

ESERCIZIO 5 (2 punti)
La seguente porzione di programma C è lecita?

```
char s1[]="Pluto"; char *s2;
strcpy(s1, "Pippo");
printf("%s", s1=s2);
```

In caso affermativo, che cosa viene stampato a video? Si giustifichi la risposta data.

Soluzione

Esercizio 1

```
list caratteri(list stringhe, int *ripetuti, int* dim){
      list new;
      char *str;
      int ripetizioni, i, trovato;
      *dim=0;
      new=emptylist();
      while(!empty(stringhe)){
            str=head(stringhe);
            ripetizioni=0;
            while((*str)!='\0'){
                  trovato=0;
                  i=1;
                  while(str[i]!='\0' && trovato==0){
                        if(str[i]==str[0]) trovato=1;
                        else i++;
                  }
                  if(trovato==1) ripetizioni++;
                  str++;
            }
            if(ripetizioni==0){
                  new=cons(head(stringhe),new);
            }
            else{
                  ripetuti[*dim]=ripetizioni;
                  *dim=*dim+1;
            }
            stringhe=tail(stringhe);
      }
      return new;
}
```

Esercizio 2

```
void split(stack *x, stack *y){
	element el;
	if(isEmptyStack(*y)) return;
	else{
		el=pop(y);
		if(el<0){
			split(x,y);
			push(el,x);
		}
		else{
			split(x,y);
			push(el,y);
		}
		return;
	}
}
```

Esercizio 3

La funzione `add()` prende in ingresso due stringhe ben formate ed un array di interi. Il ciclo `while` scandisce ogni carattere della stringa `char_v` ed ogni intero dell'array `int_v`; ogni carattere della stringa `char_v` viene sostituito con il carattere che lo precede o lo segue di un numero di posizioni uguale al valore del corrispondente intero del vettore `int_v`. In seguito la funzione controlla se il nuovo carattere è presente anche nella stringa `canc`; in caso positivo tale carattere viene eliminato. La funzione restituisce il numero di caratteri non eliminati.

La funzione `main()` richiama la funzione `add()` e stampa sullo standard output "`5 ubuvh`".

Esercizio 4

S → YNM → 3NM → 3YMNM → 32MNM → 32XMNM → 325MNM → 325XNM → 3255NM → 3255YNM → 32551NM → 32551YM → 325512M → 325512X → 3255127

Esercizio 5

La porzione di codice non è lecita dal punto di vista sintattico, poiché l'assegnamento all'interno della chiamata di funzione printf("%s", s1=s2); cerca di assegnare un nuovo valore alla variabile s1. Ciò non è possibile in quanto s1 denota l'indirizzo di un array, che è immutabile (costante).