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 A

ESERCIZIO 1 (11 punti)

Si consideri una lista di stringhe ben formate `l` e un vettore di interi `v` avente spazio sufficiente a contenere un numero di interi uguale al numero di elementi della lista `l`. Il candidato realizzi

una funzione:

```
list clean(list l, int *v, int* dim);
```

che restituisca una nuova lista con tutti e soli gli elementi della lista `l` che contengono almeno un carattere ripetuto (in ciascuna stringa ogni carattere può comparire al più due volte). Inoltre, la funzione deve inserire nel vettore `v` un intero per ciascun elemento della nuova lista restituita; tale intero rappresenta quanti sono i caratteri ripetuti nella stringa corrispondente. Infine, la funzione deve restituire tramite `dim` la dimensione logica del vettore `v`.
Ad esempio, se `l = [passare, abcd, fs, sottomarini, casa]`, la lista risultato sarà `[passare, sottomarini, casa]` (l'ordine degli elementi nella lista restituita è ininfluente) e il vettore `v = [2, 3, 1]` poiché `passare` ha due caratteri ripetuti, `sottomarini` tre e `casa` uno.

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 sull'ADT stack di interi definite durante il corso (che NON devono essere riportate), realizzare una funzione ricorsiva:

```
void divide(stack *s1, stack *s2);
```

che, dati due stack `s1` e `s2`, sposti in `s2` (stack vuoto) tutti i dati presenti in `s1` maggiori o uguali a 0. L'ordine degli elementi di `s1` deve rimanere necessariamente inalterato mentre l'ordine degli elementi di `s2` deve essere invertito. Ad esempio, se `s1 = [-2, -1, 3, 5, -7, 1, 0, -4]` dopo l'invocazione della funzione `divide()`, `s1` deve risultare `[-2, -1, -7, -4]` mentre `s2` può risultare `[0, 1, 5, 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=7;

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

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

ESERCIZIO 4 (3 punti)
Si consideri la grammatica G con scopo S e simboli terminali {a,b,c,d,x,y,z}

```
S::= DAB | EBA
A::= DA | D
B::= EB | EAB | E
D::= a | b | c | d
E::= x | y | z
```

La stringa "`xydazzb`" appartiene al linguaggio generato da tale grammatica? In caso affermativo, 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(s2, "Pippo"); s2=s1;
printf("%s", s2);
```

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

Soluzione

Esercizio 1

```
list clean(list l, int *v, int* dim){
      list new;
      char *str;
      int ripetizioni, i, trovato;
      *dim=0;
      new=emptylist();
      while(!empty(l)){
            str=head(l);
            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(l),new);
                  v[*dim]=ripetizioni;
                  *dim=*dim+1;
            }
            l=tail(l);
      }
      return new;
}
```

Esercizio 2

```
void divide(stack *s1, stack *s2){
      element el;
      if(isEmptyStack(*s1)) return;
      else{
            el=pop(s1);
            if(el>=0){
                  push(el,s2);
                  divide(s1,s2);
            }
            else{
                  divide(s1,s2);
                  push(el,s1);
            }
            return;
      }
}
```

Esercizio 3

La funzione `fun()` 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 `del`; in caso positivo tale carattere viene eliminato. La funzione restituisce il numero di caratteri eliminati.

La funzione `main()` richiama la funzione `fun()` e stampa sullo standard output "`4 fbceh`".

Esercizio 4

S → EBA → xBA → xEABA → xyABA → xyDABA → xydABA → xydDBA → xydaBA → xydaEBA → xydazBA → xydazEA → xydazzA → xydazzD → xydazzb

Esercizio 5

Nonostante dal punto di vista meramente sintattico il codice sia corretto e compilabile, da un punto di vista logico la porzione di codice non è corretta, in quanto l'istruzione strcpy(s2, "Pippo"); cerca di scrivere dati in una zona di memoria puntata da s2 che, non essendo stata ancora inizializzata ad alcun valore sensato, è da considerarsi casuale. Quindi non esiste nessuna garanzia che la zona di memoria puntata da s2 sia accessibile o s2 sia un indirizzo valido.