Fondamenti di Informatica T-1 (A.A. 2008/2009) - Ingegneria Informatica
Prof.ssa Mello
Prova Parziale d'Esame di Lunedì 22 Dicembre 2008 – durata 1h
Compito B

ESERCIZIO 1 (7 punti)

Si supponga di avere a disposizione, già definito, l'ADT lista per interi (denominato `list`, con relative primitive). Il candidato definisca una funzione iterativa:

```
list ripetuti(list l1);
```

La funzione deve restituire una nuova lista contenente i valori di `l1` che compaiono ripetuti più volte (cioè almeno due volte), eventualmente non in maniera ordinata. La nuova lista non deve contenere elementi ripetuti. La funzione deve essere implementata accedendo alle liste tramite la notazione a puntatore, e senza fare ricorso alle primitive dell'ADT.

Al fine di semplificare la funzione di cui sopra, il candidato realizzi una funzione *ricorsiva*:

```
int conta(int el, list l1);
```

che conta quante volte l'elemento `el` compare in `l1`. Tale funzione deve essere implementata utilizzando le sole primitive dell'ADT lista.

Ad esempio, se `l1 = {1, 3, 2, 4, 2, 1, 3, 1, 1, 2}`, la funzione restituirà una lista `{1, 3, 2}`, poiché il valore 1 compare 4 volte, il valore 3 compare 2 volte ed il valore 2 compare 3 volte.

ESERCIZIO 2 (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)

```
#include <stdio.h>
int dim = 4;

char * gambero(char v[], int dim, int step){
    int i;
    char * result = (char*) malloc(sizeof(char) * (dim+1));

    for(i=0; i<dim; i++)
        result[(i+step)%dim] = v[i%dim];
      result[dim] = '\0';
    return result;
}

int main(){
      char msg[] = "Bologna";
      char * temp = msg;
      int dim=0;
      for ( ; *temp!='\0'; ) {
            temp+=1;;
            dim+=1;
      }
      temp = gambero(msg, dim, 3);
      printf("%s %d", temp, dim);
      return (0);
}
```

ESERCIZIO 3 (2 punti)

Si consideri la grammatica G con scopo S, simboli non terminali {V, W, X, Y, Z} e simboli terminali {a, b, c, d, e, 1, 2, 3, 4}:

```
S::= VZ | WZ
V::= WXV | WYZ
W::= XYX | ZYZ
X::= a | b | c
Y::= d | e
Z::= 1 |2 | 3 | 4
```

La stringa "aebcadce23" appartiene al linguaggio generato da tale grammatica?
In caso affermativo, se ne mostri la derivazione left-most.

Soluzione

ESERCIZIO 1

```
int conta(int el, list l1) {
      if (empty(l1))
            return 0;
      else
            if (head(l1) == el)
                  return 1 + conta(el, tail(l1));
            else
                  return conta(el, tail(l1));
}

list ripetuti(list l1) {
      list result = NULL;
      list temp;
      int el;

      while (l1 != NULL) {
            el = l1->value;
            if (conta(el, l1) > 1 && conta(el, result)==0 ) {
                  temp = (item*) malloc(sizeof(item));
                  temp ->value = el;
                  temp ->next = result;
                  result = temp;
            }
            l1 = l1->next;
      }
      return result;
}
```

ESERCIZIO 2

Il programma è corretto sintatticamente e la sua esecuzione produce la stampa:
gnaBolo 7

La funzione `main()` dapprima calcola la lunghezza della stringa `msg`, e poi invoca la funzione `gambero(…)`.
Tale funzione alloca spazio sufficiente per contenere la stringa passata tramite il parametro `v`, più uno spazio ulteriore per il terminatore di stringa. Il ciclo `for` successivo provvede a copiare i caratteri di `v` nello spazio di memoria appena allocato, ma operando uno "shift" a destra di `step` posizioni dei caratteri, avendo cura di settare comunque il terminatore di stringa.
La funzione `main()` infine stampa a video la stringa ottenuta dalla funzione `gambero(…)`, e la lunghezza della stringa.

ESERCIZIO 3

La frase appartiene al linguaggio. In particolare, la si può ottenere tramite la seguente derivazione left-most:
S -> VZ -> WXVZ -> XYXXVZ -> aYXXVZ -> aeXXVZ -> aebXVZ -> aebcVZ -> aebcWYZZ -> aebcXYXYZZ -> aebcaYXYZZ -> aebcadXYZZ -> aebcadcYZZ -> aebcadceZZ -> aebcadce2Z -> aebcadce23