PROGRAMMAZIONE I (A,B) - a.a. 2016-17
Secondo Appello – 9 febbraio 2017

Esercizio 1

Si scriva una funzione **C** che, dato un array a di dimensione dim, restituisce il valore di verità della seguente formula

$$\forall i \in [2, dim).a[i] = (\Sigma j : j \in [1, i-1].a[j]) + (\Sigma k : k \in [0, i-2].a[k])$$

SOLUZIONE Un esempio di array che verifica la formula data è il seguente

```
int a[6] = {1,2,3,8,19,46};
```

Seguendo la struttura delle formula logica, una prima soluzione prevede l'uso di due cicli da inserire all'interno di un ciclo più esterno. In modo modulare, questo risultato può essere ottenuto definendo una funzione ausiliaria che calcola la somma degli elementi di una porzione dell'array.

```
int somma (int a[], int init, int end) {
  int somma=0;
  for (int k=init; k<=end; k++)
    somma+=a[k];
  return somma;
}

int check (int a[], int dim) {
  for (int i=2; i<dim; i++)
    if (a[i]!=(somma(a,1,i-1)+somma(a,0,i-2)))
      return 0;
  return 1;
}
```

Ragionando sulla formula si può osservare che in realtà il calcolo del valore che deve assumere ogni elemento è riconducibile al valore dell'elemento che lo precede come segue

$$a[i] = a[i-1] + a[i-1] + a[i-2] \qquad i > 2$$

Questa osservazione porta alla seguente soluzione

```
int check (int a[], int dim) {
  if (dim>2 && a[2]!=(a[0]+a[1]))
    return 0;
  for (int i=3; i<dim; i++)
    if (a[i]!=(a[i-1]+a[i-2]+a[i-1]))
      return 0;
  return 1;
}
```

Una variante della soluzione precedente evita di trattare il caso $i = 2$ come separato

```
int check (int a[], int dim) {
  int prec=0;
  for (int i=2; i<dim; i++) {
    if (a[i]!=(prec+a[i-2]+a[i-1]))
      return 0;
    prec=a[i];
  }
  return 1;
}
```

Esercizio 2

Dato il seguente linguaggio sull'alfabeto $\Lambda = \{a, b, c, d, e\}$

$$\mathcal{L} = \{aa(bb)^n cc(dd)^m ee \mid n > m > 0\}$$

si verifichi se il linguaggio è regolare o meno (fornendo una opportuna dimostrazione) e si definisca una grammatica che lo genera.

SOLUZIONE Il legame fra gli esponenti di b e di c dovrebbe far sospettare che il linguaggio non sia regolare.

Per dimostrarlo applicando il *pumping lemma*, dato un qualunque numero naturale n, prendiamo la stringa $w = aab^{2n+2}ccd^{2n}ee$. Le possibili decomposizioni $w = xyz$ sono:

- $x = \epsilon$, $y = aab^t$ e $z = b^{2n+2-t}ccd^{2n}ee$, con $0 < t \leq n - 2$
- $x = a$, $y = ab^t$ e $z = b^{2n+2-t}ccd^{2n}ee$, con $0 < t \leq n - 2$
- $x = aab^s$, $y = b^t$ e $z = b^{2n+2-t-s}ccd^{2n}ee$, con $0 \leq s < n - 2$ e $0 < t \leq n - 2 - s$

Nei primi due casi abbiamo che $xy^0z \notin \mathcal{L}$ poiché tale stringa contiene solo una o zero simboli a iniziali. Nel terzo caso, $xy^0z \notin \mathcal{L}$ dato che: se t è dispari la stringa contiene un numero dispari di simboli b, e se t è pari (e $t > 0$) abbiamo $2n + 2 - t \leq 2n$.

Una grammatica che genera il linguaggio è la seguente:

```
S -> aaBCee
B -> bb | bbB
C -> bbCdd | bbccdd
```

Esercizio 3

Si suppongano predefiniti i tipi

```
struct el {int info; struct el *next;};
typedef struct el ElementoDiLista;
typedef ElementoDiLista* ListaDiElementi;
```

Si scriva in **C** una procedura che, presi una lista e un numero naturale n, elimina gli ultimi n elementi della lista e inserisce in coda (al loro posto) un nuovo elemento il cui valore è pari alla somma dei valori contenuti negli elementi rimossi. Se la lista contiene meno di n elementi, l'operazione viene eseguita usando tutti gli elementi contenuti nella lista. Se nessun elemento viene rimosso (perché $n = 0$ o perché la lista è vuota) si dovrà comunque inserire in coda un nuovo elemento contenente il valore 0.

SOLUZIONE Una prima soluzione si basa sull'utilizzo delle funzioni di cancellazione e inserimento in coda (viste a lezione) in cui la prima funzione è modificata al fine di farle restituire il valore contenuto nell'elemento cancellato.

```
void elimina (ListaDiElementi *l, int n)
{
  int somma=0;
  while (n>0)
  {
    n--;
    somma+=cancC(l);
  }
  addC(l, somma);
}

int cancC (ListaDiElementi *l)
{
  int val = 0;
  if (*l!=NULL)
  {
    if ((*l)->next==NULL)
    {
      val=(*l)->info;
      free(*l);
      *l=NULL;
    }
    else
    {
      ListaDiElementi corr = (*l)->next;
      ListaDiElementi prec = *l;
      while (corr->next!=NULL)
       {
        prec=corr;
        corr=corr->next;
      }
```

```
      val=corr->info;
      free(corr);
      prec->next=NULL;
    }
  }
  return val;
}

void addC (ListaDiElementi *l, int x)
{
  ListaDiElementi new = malloc(sizeof(ElementoDiLista));
  new->info = x;
  new->next = NULL;
  if (*l==NULL)
    *l = new;
  else
  {
    ListaDiElementi corr = *l;
    while (corr->next!=NULL)
      corr = corr->next;
    corr->next=new;
  }
}
```

La soluzione precedente scandisce la lista n volte. Una soluzione più efficiente consiste nel calcolare la lunghezza della lista, sottrarre n a tale valore (se possibile), ed usare il risultato per individuare il primo elemento della lista che dovrà essere cancellato. In questo modo le cancellazioni potranno essere eseguite sequenzialmente, senza dover ogni volta ripartire con un puntatore dall'inizio della lista.

```
void elimina (ListaDiElementi *l, int n)
{
  ListaDiElementi corr = *l;
  ListaDiElementi prec = NULL;

  int lunghezza = length(*l);
  int somma = 0;

  ListaDiElementi new = malloc(sizeof(ElementoDiLista));

  if (lunghezza>n)
  {
    while (lunghezza>n)
    {
      prec=corr;
      corr=corr->next;
      lunghezza--;
    }
  }
  somma = eliminasomma(corr);
```

```
  new->info=somma;
  new->next=NULL;
  if (prec!=NULL)
    prec->next=new;
  else
    *l=new;
}

int eliminasomma (ListaDiElementi l)
{
  int somma=0;
  ListaDiElementi tmp;
  while (l!=NULL)
  {
    somma+=l->info;
    tmp=l;
    l=l->next;
    free(tmp);
  }
  return somma;
}

int length (ListaDiElementi l)
{
  int lunghezza=0;
  while (l!=NULL)
  {
    lunghezza++;
    l=l->next;
  }
  return lunghezza;
}
```

Esercizio 4

Si definisca in CAML, senza usare la ricorsione esplicita, una funzione

`media : int list -> int`

che, data una lista `lis` di interi non vuota, restituisce la media artimetica dei valori in essa contenuti. La funzione non è definita nel caso di lista vuota.

SOLUZIONE Ecco di seguito tre possibili soluzioni

```
let media lis =
    match lis with
      x::xs -> let f x (y1,y2) = (y1+x,y2+1)
                 in
               let (somma,cont) = foldr f (0,0) lis
                 in
               somma/cont;;

let media lis =
    match lis with
      x::xs -> let f x (y1,y2) = (y1+x,y2+1)
                 in
               let dividi (x,y) = x/y
                 in
               dividi (foldr f (0,0) lis);;

let media lis =
    match lis with
      x::xs -> let f x y = x+y
                 in
               let g x y = 1+y
                 in
               (foldr f 0 lis) / (foldr g 0 lis);;
```