Architettura degli Elaboratori – Anno Accademico 2021 – 2022
Appello 23 giugno 2022 – Corsi A e B

Si scrivano due funzioni in codice assembler:

- una funzione **int ismatch(char *s)** che controlla se la stringa s inizia con una stringa nella forma ab^+a e restituisce **0** se questo non è vero o **k**, numero di caratteri nella parte $a(b+)a$ della stringa. La stringa è una stringa di caratteri ASCII terminata da un **NULL** (codice 0). Il codice ASCII della lettera a è *97*.
  Per esempio, la funzione chiamata con "abbaaba" dovrebbe restituire 4, con "aaab" 0, con "baba" 0, con "aaba" 0, con "abaaa" 3;
- una funzione **int * occurrences(char *a[], int n)** che restituisce un vettore di n interi, ciascuno dei quali conta il numero di occorrenze ***distinte*** (cioè non sovrapposte) di stringhe ab^+a nella posizione corrispondente del vettore a. n è la lunghezza del vettore a. Ogni a[j] è un puntatore a una stringa.
  Per esempio, la funzione chiamata con il vettore {"abaaa", "abba", "baaba", "aaba"} e l'interno 4 dovrebbe restituire il puntatore a un vettore contenente {3,4,0,0}.

Per testare il codice potete utilizzare il codice C

```
#include <stdio.h>
#include <stdlib.h>
extern int ismatch(char * s);
extern int * occurrences(char *a[], int n);

int main(int argc, char ** argv) {
  char *a[] = {"abba", "baa", "baa", "babab",
"ababaabbba","acdeabbbaaaaabcccabaaabbbbbababababbaabbaa"};
  int i; int * r = occurrences(a,6);
  for(i = 0; i<6; i++)
    printf("Trovate %d occorrenze in %s \n", r[i], a[i]);
  return(0);
}
```

eseguendo il quale si dovrebbe ottenere:

```
Trovate 1 occorrenze in abba
Trovate 0 occorrenze in baa
Trovate 0 occorrenze in baa
Trovate 1 occorrenze in babab
Trovate 2 occorrenze in ababaabbba
Trovate 5 occorrenze in acdeabbbaaaaabcccabaaabbbbbababababbaabbaa
```

Il codice dovrebbe contare le occorrenze "distinte", quindi **ababaabbba** ha due sole occorrenze **aba**ba**abbba,** in quanto la sequenza che comincia con la seconda a non è riconosciuta, essendo la seconda a ultima parte della prima stringa riconosciuta.

Soluzione

occurences.s

```
.text
.global occurrences
.type occurrences, %function

occurrences:      @ r0 VETTORE DI PUNTATORI A CARATTERI a[]
@ r1 LUNGHEZZA DEL VETTORE

push {r4-r9,lr}         @ salva registri
mov r4, r0        @ a
mov r5, r1        @ n
mov r6, #0        @ i
mov r0, r1, lsl #2      @ n + sizeof int
bl malloc
mov r7, r0        @ indirizzo di r[]
for:  cmp r6, r5        @ i == n ?
beq fine          @ in caso termina
mov r8, #0        @ count = 0
ldr r9, [r4,r6,lsl #2]        @ p = a[i]
while:      ldrb r0, [r9]           @ *p
cmp r0, #0        @ *p == null
beq finew         @ allora termina while
mov r0, r9        @ ismatch(p)
bl ismatch
cmp r0, #0        @ controlla risultato
addne r8, r8, #1  @ se trovato incrementa count
addne r9, r9, r0  @ e sposta puntatore a fine str
addeq r9, r9, #1  @ altrimenti incrementa p
b while                 @ altra iterazione
finew:      str r8, [r7, r6, lsl #2] @ r[i] = count
add r6, r6, #1          @ i++
b for                   @ prossima iterazione
fine:       mov r0, r7        @ return indirizzo del vettore r
pop {r4-r9,pc}          @ ripristina registri e LR -> PC (ret)
```

IsMatch.s

```
        .text
        .global ismatch
        .type ismatch, %function

        @ r0 -> indirizzo della stringa

ismatch:mov r1, #0      @ contatore caratteri
        ldrb r3, [r0]     @ primo carattere
        cmp r3, #97 @ controlla se fosse una a
        bne nontrovato    @ se non lo fosse, restituisci 0
        add r1, r1, #1    @ count ++
        add r0, r0, #1  @ prossimo catattere
        add r1, r1, #1  @ +1 carattere
        ldrb r3, [r0]
        cmp r3, #98 @ se non e' una b
        bne nontrovato
bbb:  ldrb r3, [r0]
        cmp r3, #98
        addeq r0, r0, #1
        addeq r1, r1, #1
        beq bbb
        cmp r3, #97 @ controlla ultima a
        bne nontrovato
        mov r0, r1  @ restituisci count
        mov pc, lr
nontrovato:
        movne r0, #0
        movne pc, lr
```