Esercizio 1

```
let const = 10;;

let myfun l x y = match l with
  [] -> x
  | a::ls -> if a > 0 then x+y else y-x;;

 let checkAndApply l (f:int ->int) =
  let rec aux l f = match l with
    [] -> const
   | elem::ls -> if elem > 0 then f const else aux ls f
   in aux l f;;

let ls1 = [100;1000];;

let ls2 = [1;2;3;4];;

let const = 5;;

checkAndApply ls2 ((myfun ls1) const);;
```

Si mostri la struttura della pila dei record di attivazione al momento dell'invocazione della funzione identificata dal parametro formale f.

Esercizio 2

```
let init = 5;;

let startAt x =
   let incrementBy y = x + y
   in incrementBy;;

let rec applytwice f l = match l with
  [] -> []
  | x::xs -> f x x :: applytwice f xs;;

applytwice startAt [init+6; init+8];;
```

Si simuli la valutazione del programma mostrando la struttura della pila dei record di attivazione.

S1	SL=M	CL=M
	const	10
S2	SL = S1	CL = S1
	myfun	M1
S3	SL = S2	CL = S2
	checkApply	M2
S4	SL = S3	CL = S3
	ls1	[100;1000]
	ls2	[1;2;3;4]
	const	5
S5	SL= S1	CL = S4
	l	[100;1000]
	x	5
	Resull	M3
	Retained	
S6	SL = S2	CL = S4
	l	[1;2;3;4]
	f	M3
S7	SL= S6	CL=S6
	aux	M4
S8	SL=S5	CL=S7
	y	5
	Result	x+y

M1	<code, S1>
M2	<code, S2>
M3	<fun y ->x+y, S5>
M4	<code, S7>

I parametri vengono valutati nell'ambiente del chiamante.
Eseguo parzialmente myfun che è un parametro di checkAndApply.
(myfun ls1 const)
myfun è memorizzata in M1 e ha static link S1.
Passaggio dei parametri: l = ls1 = [100; 1000], x = const = 5.
Il risultato è una funzione (fun y -> x + y) e la memorizzo in M3.
Il blocco S5 è "retained", cioè non deve essere cancellato perché serve per gestire la funzione anonima memorizzata in M3 quando sarà invocata.

(checkAndApply ls2 f)
checkAndApply è memorizzata in M2 e ha static link S2.
Il control link punta a S4 e non a S5 perchè S5 è retained, non serve più per la valutazione dell'esecuzione ma solo per la valutazione di ambiente.
Passaggio dei parametri: l = ls2 = [1; 2; 3; 4], f = (fun y -> x + y) = M3.

In checkAndApply c'è un blocco più interno che contiene la funzione ricorsiva aux che memorizzo in M4. Essendo ricorsiva il puntatore della chiusura punta all'ambiente dove è stata dichiarata cioè S7.
Per sapere qual è lo static link di questo blocco vado a vedere qual è il puntatore di ambiente della chiusura di checkAndApply, S2.

vengono saltati un po' di passi

Invocazione del parametro formale f in aux.
f è memorizzata in M3 e ha static link S5.

Il risultato è x + y = 5 + 5 = 10.

MEM ADDR	
M1	<code, S1>
M2	<code, S3>
M3	<code,S5>
M4	<code, S7>

STACK ADDR		
S1	init	5
S2	SL= S1	CL=S1
	startAt	M1
S3	SL = S2	CL = S2
	applytwice	M2
S4	SL = S3	CL=S3
	f	M1
	l	[11;13]
	x	11
	xs	[13]
	Res	
	Tmp	22
S5	SL=S1	CL=S4
	x	11
	incrementBy	M3
	y	11
	Result	22
S6	SL = S3	CL=S4
	f	M1
	l	[13]
	x	13
	xs	[]
	Res	[36]
	Tmp	36
S7	SL=S1	CL=S6
	x	13
	incrementBy	M4
	y	13
	Result	26

Pass-param [int+6;init+8]= [11;13]

Popped

Popped