**LABORATORIO 5/03/15**

let x : int = 5;; #definisco un intero e ci assegno il valore 5

x;; #stampo la variabile

let y : float = 5;; #definisco una variabile di tipo float

let square x = x \* x;; #creo una funzione che dato un intero mi restituisce il valore elevato alla 2

es:

>square 10;;

val it : int 100

let square = fun x -> x \*x;; #è la stessa cosa

**ESERCIZI 9/03/15**

**ESERCIZI FUNZIONI RICORSIVE**

1)

• let rec mcd = function

|(0,n)->n

|(m,n)->mcd(n%m,m);;

• let simplify (m,n)=(m/mcd(m,n),n/mcd(m,n));;

• let rec sum1 = function

|0->0

|n->n+sum1(n-1);;

•

let rec sum2(m:int,n) =

let p = n-m

match n with

|p->p

//|m->0

|x-> sum2(m,x-1);;

let rec sum2 (m,n) =

if n=m then m

elif n<m then 0

else n+sum2(m,n-1);;

• let rec fibo = function

|0->0

|1 | 2->1

|n->fibo(n-1)+fibo(n-2)

ESERCIZIO COSTO

• let prA cod =

match cod with

| ”cod1” -> 20.0

| ”cod2” -> 50.50

| \_ -> 0.0;;

let prB cod =

match cod with

| ”cod1” -> 40.0

| ”cod2” -> 100.50

| \_ -> 0.0;;

let scA cod =

match cod with

| ”cod1” -> 10.0

| ”cod2” -> 0.0

| \_ -> 0.0;;

ESERCIZIO LEZIONE LAB 6

ES 1

let sommaArea (f1,f2)=

match areaOpt f1, areaOpt f2 with

|Some a1, Some a2-> Some (a1+a2)

| \_, None | None, \_ ->None;;

ES 2

b)

let rec lastOpt ls=

match ls with

|[]->None

|[x]->Some x

|x0::xs->lastOpt xs;;

c)

let rec catOpt ls=

match ls with

|[]->[]

|x0::xs ->

match x0 with

| Some y -> y::catOpt xs

| None -> catOpt xs;;

d)

let rec mynth (ls,n) =

match (ls,n) with

| ([],->None

|