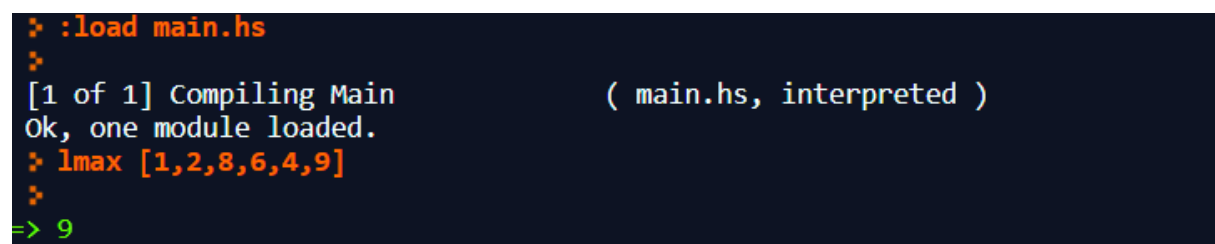


Haskell

Code Snippets and Screenshot:

1. Write a program that finds the maximum of a list of numbers.

```
lmax :: [Int] -> Int
lmax (x:[]) = x
lmax (x:xs) | (lmax xs) > x = lmax xs
            | otherwise     = x
```



```
> :load main.hs
>
[1 of 1] Compiling Main                ( main.hs, interpreted )
Ok, one module loaded.
> lmax [1,2,8,6,4,9]
>
=> 9
```

2. Write a program that succeeds if the intersection of two given list parameters is empty.

```
intersect (x:xs) (y:ys)
```

```
  | x == y  = x : intersect xs ys
```

```
  | x < y   = intersect xs (y:ys)
```

```
  | x > y   = intersect (x:xs) ys
```

```
intersect x [] = []
```

```
intersect [] y = []
```

```
length_of_list [] = 0
```

```
length_of_list (x:xs) = 1 + length_of_list xs
```

```
intersection xs ys =
```

```
  if length_of_list (intersect (sort xs) (sort ys)) == 0 then True
```

```
  else False
```

```
> intersection [1,2,3] [5,6,7]
>
=> True
> intersection [1,2,3] [1,5,6,7]
>
=> False
> 
```

3. Write a program that returns a list containing the union of the elements of two given lists.

```
add_lists (x:xs) (y:ys) =  
  if x == y then x: union xs ys  
  else if x > y then y: union (x:xs) ys  
  else x: union xs (y:ys)  
add_lists [] ys = ys  
add_lists xs [] = xs
```

```
duplicate [] = []  
duplicate [x] = [x]  
duplicate (x:xs) =  
  if x == y then duplicate xs  
  else x:duplicate xs  
  where y = head(xs)
```

```
union (x) (y) = add_lists (duplicate (sort x)) (duplicate (sort y))
```

```
❖ union [1,2,3,4,22,1] [2,3,1,1]  
❖  
> [1,2,3,4,22]  
❖
```

4. Write a program that returns the final element of a list

final (x:[]) = x

final (x:xs) = final xs

```
> :load main.hs
>
[1 of 1] Compiling Main                ( main.hs, interpreted )
Ok, one module loaded.
> final [1,2,3,4,22,1]
>
=> 1
> █
```

Prolog

Code Snippets and Screenshot:

1. Write a program that finds the maximum of a list of numbers.

`maxe([X],X).`

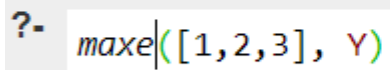
`maxe([X|Y],X):- maxe(Y,P), X >= P.`

`maxe([X|Y],Q):- maxe(Y,Q), Q > X.`

A screenshot of a Prolog code editor. It shows a query icon (a gear with a smiley face) followed by the text `maxe([1,2,3], Y)`.

`maxe([1,2,3], Y)`

`Y = 3`

A screenshot of a Prolog code editor. It shows a query icon (a question mark) followed by the text `maxe([1,2,3], Y)`.

`?- maxe([1,2,3], Y)`

2. Write a program that succeeds if the intersection of two given list parameters is empty.

li([], _, []).

li([Head|L1tail], L2, L3) :-

memberchk(Head, L2),

!,

L3 = [Head|L3tail],

li(L1tail, L2, L3tail).

li([_ | L1tail], L2, L3) :-

li(L1tail, L2, L3).



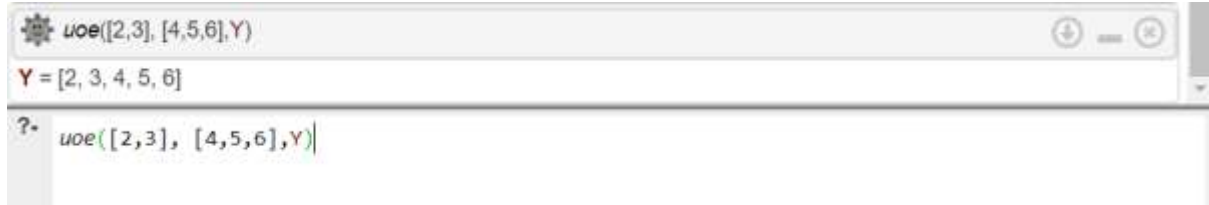
The image shows a Prolog interpreter window with a title bar containing a gear icon, the text "li([2,3], [1,4,5,6],Y)", and standard window controls (down arrow, minus, close). The main area displays the query "Y = []". Below this is another window with the title bar "li([2,3], [1,4,5,6,2],Y)" and the result "Y = [2]". At the bottom, a prompt "?- li([2,3], [1,4,5,6,2],Y)" is shown.

3. Write a program that returns a list containing the union of the elements of two given lists.

uoel([],X,X).

uoel([X|Y],P,Q):- member(X,P),!,uoel(Y,P,Q).

uoel([X|Y],P,[X|Q]):- uoel(Y,P,Q).



4. Write a program that returns the final element of a list

`final([X],X).`

`final([Y|X],H):- final(X,H).`

