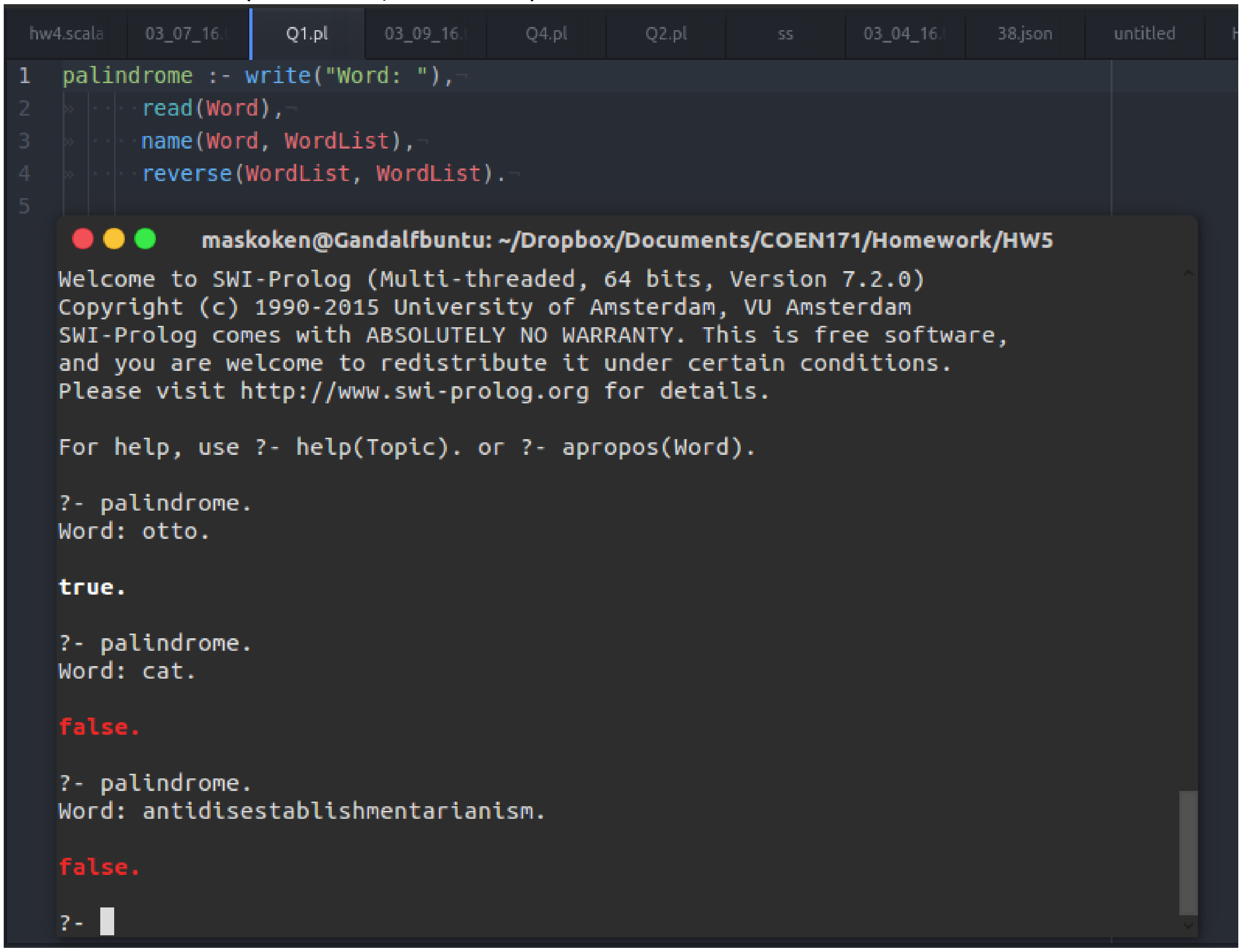
Matthew Koken
COEN 171 Programming Languages
Homework 5

All questions answered using swi-prolog on 15.10

1. Write a Prolog program to check whether a word is a palindrome, and write a program to ask you for words and tell you if your input is palindromic.

```
        palindrome :- write("Word: "),
            read(Word),
            name(Word, WordList),
            reverse(WordList, WordList). %check if the reversed version is the same
```



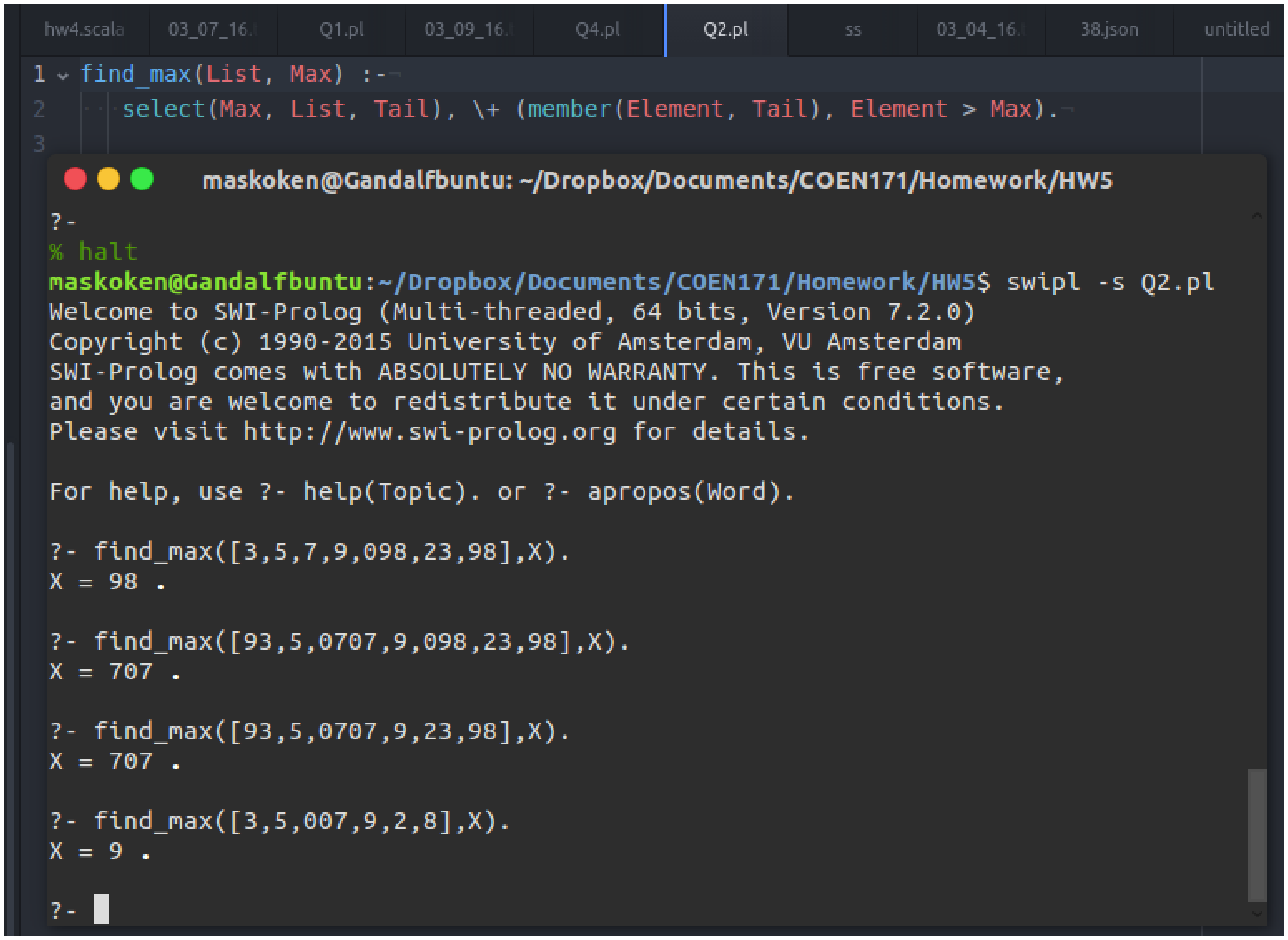
```
palindrome :- write("Word: "),
    read(Word),
    name(Word, WordList),
    reverse(WordList, WordList).
```

maskoken@Gandalfbuntu: ~/Dropbox/Documents/COEN171/Homework/HW5

```
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.0)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- palindrome.
Word: otto.

true.

?- palindrome.
Word: cat.

false.

?- palindrome.
Word: antidisestablishmentarianism.

false.

?-
```

2. Write a Prolog program that finds the max of a list

```
        find_max(List, Max) :-
          select(Max, List, Tail), \+ (member(Element, Tail), Element > Max).
```



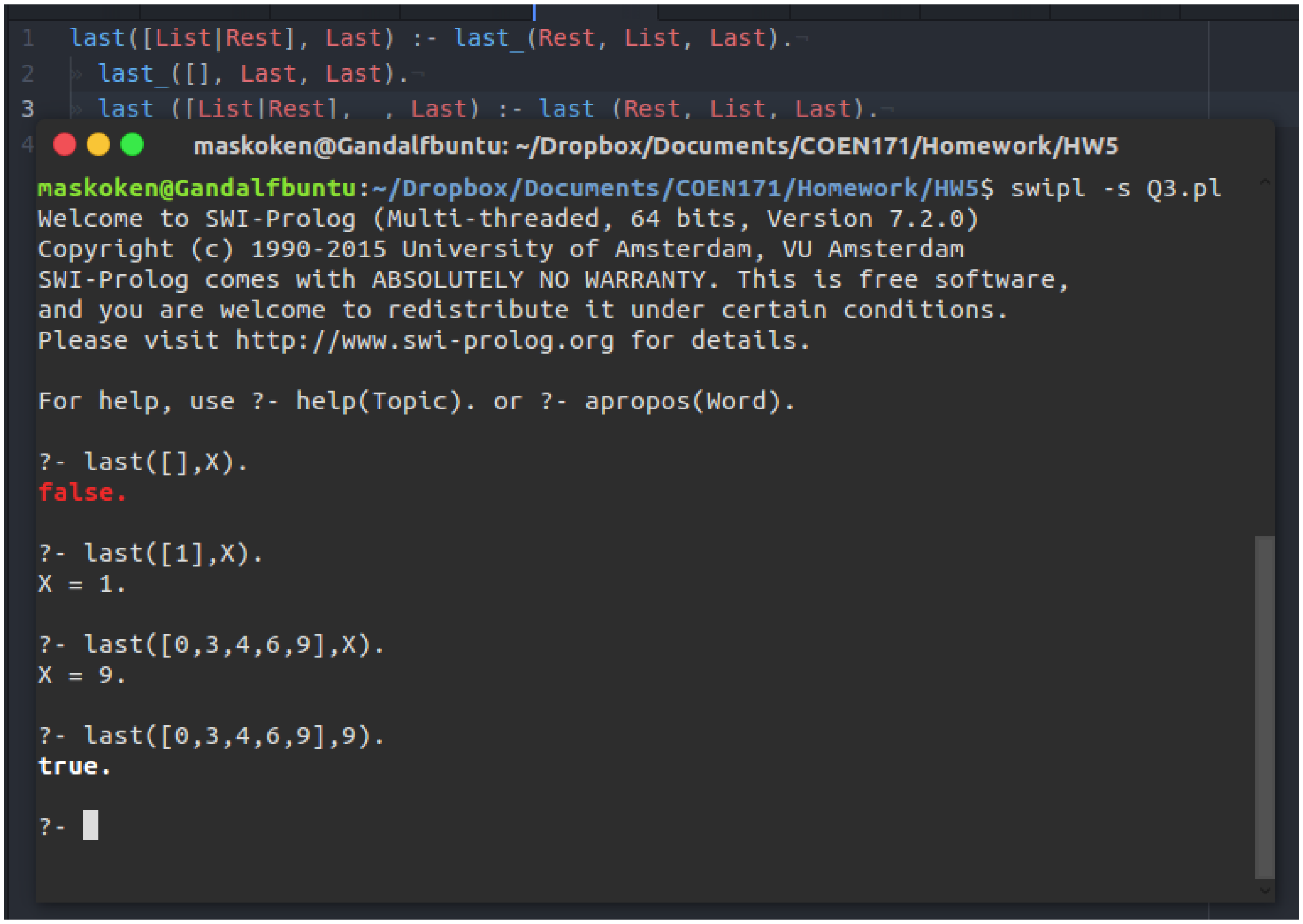
```
find_max(List, Max) :-
    select(Max, List, Tail), \+ (member(Element, Tail), Element > Max).
```

maskoken@Gandalfbuntu: ~/Dropbox/Documents/COEN171/Homework/HW5

```
?-
% halt
maskoken@Gandalfbuntu:~/Dropbox/Documents/COEN171/Homework/HW5$ swipl -s Q2.pl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.0)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- find_max([3,5,7,9,098,23,98],X).
X = 98 .

?- find_max([93,5,0707,9,098,23,98],X).
X = 707 .

?- find_max([93,5,0707,9,23,98],X).
X = 707 .

?- find_max([3,5,007,9,2,8],X).
X = 9 .

?-
```

3. Write a Prolog program that returns the last element of a list

```
%move through rest until there is only the last
last([List|Rest], Last) :- last_(Rest, List, Last).
last_([], Last, Last).
last_([List|Rest], _, Last) :- last_(Rest, List, Last).
```

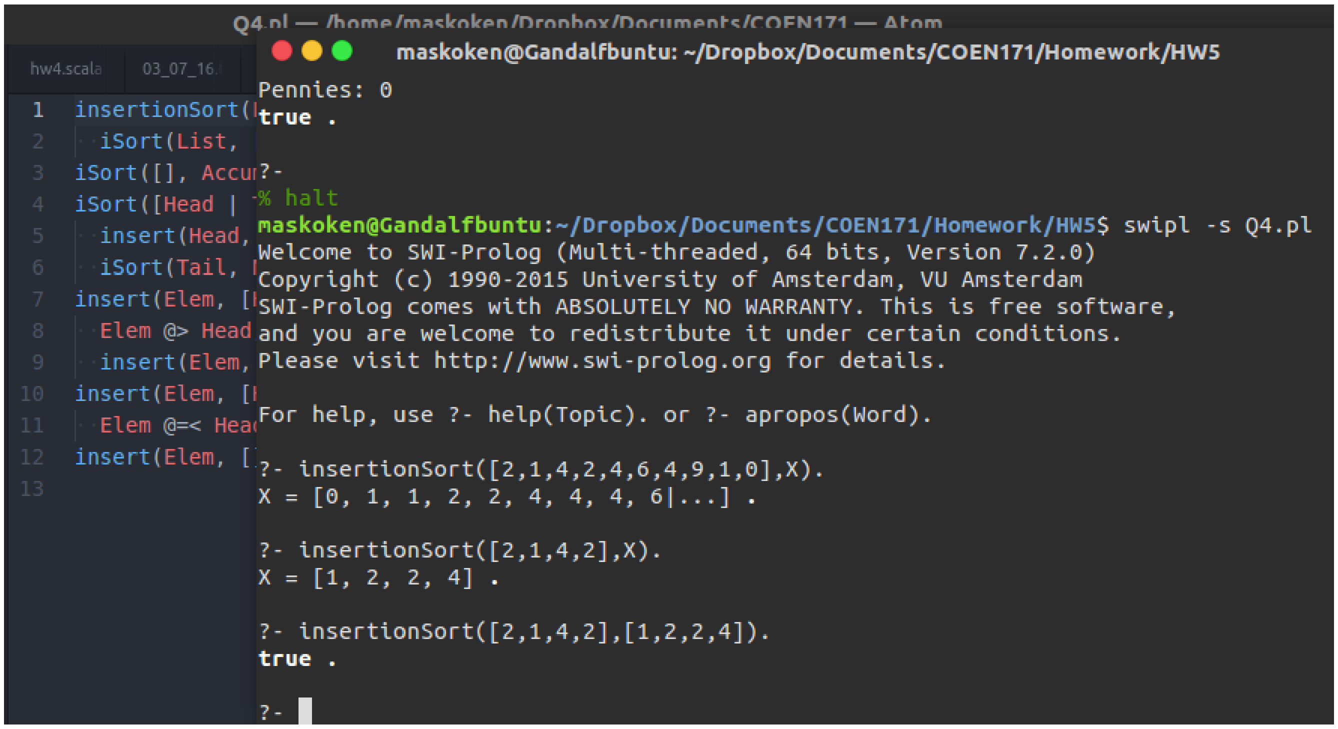


```
last([List|Rest], Last) :- last_(Rest, List, Last).
    last_([], Last, Last).
    last_([List|Rest], _, Last) :- last_(Rest, List, Last).
```

```
maskoken@Gandalfbuntu: ~/Dropbox/Documents/COEN171/Homework/HW5
maskoken@Gandalfbuntu:~/Dropbox/Documents/COEN171/Homework/HW5$ swipl -s Q3.pl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.0)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- last([],X).
false.

?- last([1],X).
X = 1.

?- last([0,3,4,6,9],X).
X = 9.

?- last([0,3,4,6,9],9).
true.

?- 
```

4. Write a Prolog program that implements insertion sort

```
insertionSort(List, SortedList) :-
        iSort(List, [], SortedList).
iSort([], Accumulator, Accumulator).
iSort([Head | Tail], Accumulator, SortedList) :-
        insert(Head, Accumulator, NewAccumulator),
        iSort(Tail, NewAccumulator, SortedList).
insert(Elem, [Head | Tail], [Head | NewTail]) :-
        Elem @> Head,
        insert(Elem, Tail, NewTail).
insert(Elem, [Head | Tail], [Elem, Head | Tail]) :-
        Elem @=< Head.
insert(Elem, [] , [Elem])
```



Q4.pl — /home/maskoken/Dropbox/Documents/COEN171 — Atom

hw4.scala 03_07_16.

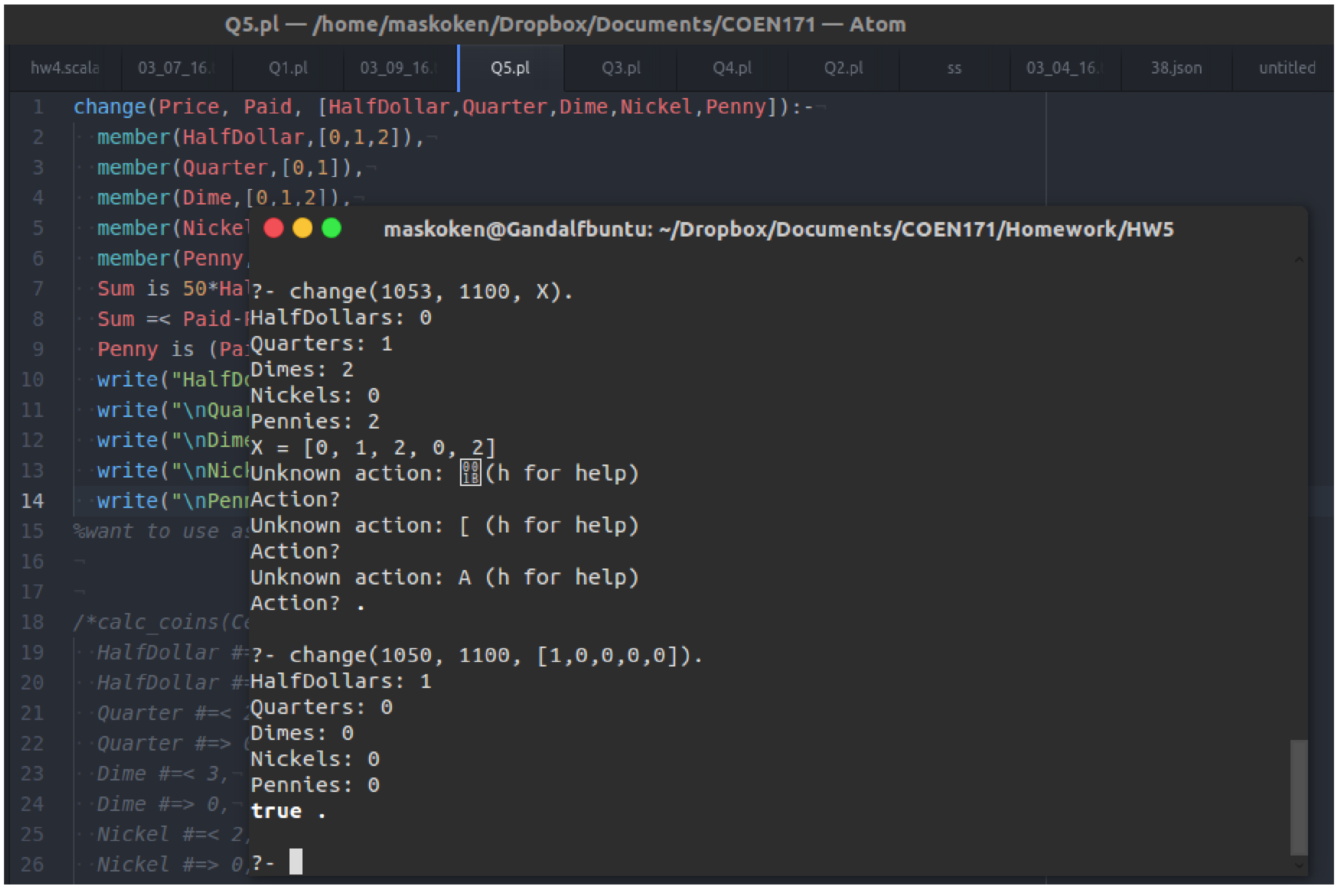
```
insertionSort(
  iSort(List,
iSort([], Accu
iSort([Head |
  insert(Head,
  iSort(Tail,
insert(Elem, [
  Elem @> Head
  insert(Elem,
insert(Elem, [
  Elem @=< Hea
insert(Elem, [
```

maskoken@Gandalfbuntu: ~/Dropbox/Documents/COEN171/Homework/HW5

```
Pennies: 0
true .

?-
% halt
maskoken@Gandalfbuntu:~/Dropbox/Documents/COEN171/Homework/HW5$ swipl -s Q4.pl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.0)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- insertionSort([2,1,4,2,4,6,4,9,1,0],X).
X = [0, 1, 1, 2, 2, 4, 4, 4, 6|...] .

?- insertionSort([2,1,4,2],X).
X = [1, 2, 2, 4] .

?- insertionSort([2,1,4,2],[1,2,2,4]).
true .

?-
```

5. Write a Prolog program that checks or generates change adding up to a dollar consisting of half-dollars, quarters, dimes, nickels, and pennies

```
%price given in cents to keep with integers → prolog doesn't like float that much
change(Price, Paid, [HalfDollar,Quarter,Dime,Nickel,Penny]):-
  member(HalfDollar,[0,1,2]),
  member(Quarter,[0,1]),
  member(Dime,[0,1,2]),
  member(Nickel,[0,1]),
  member(Penny,[0,1,2,3,4]),
  Sum is 50*HalfDollar + 25*Quarter + 10*Dime + 5*Nickel,
  Sum =< Paid-Price,
  Penny is (Paid - Price) - Sum,
  write("HalfDollars: "),write(HalfDollar),
  write("\nQuarters: "),write(Quarter),
  write("\nDimes: "),write(Dime),
  write("\nNickels: "),write(Nickel),
  write("\nPennies: "),write(Penny).
%want to use as many of the largest coins as possible
```


Q5.pl — /home/maskoken/Dropbox/Documents/COEN171 — Atom
hw4.scala
03_07_16.
Q1.pl
03_09_16.
Q5.pl
Q3.pl
Q4.pl
Q2.pl
ss
03_04_16.
38.json
untitled
maskoken@Gandalfbuntu: ~/Dropbox/Documents/COEN171/Homework/HW5

```
change(Price, Paid, [HalfDollar,Quarter,Dime,Nickel,Penny]):-
  member(HalfDollar,[0,1,2]),
  member(Quarter,[0,1]),
  member(Dime,[0,1,2]),
  member(Nickel
  member(Penny,
  Sum is 50*Ha
  Sum =< Paid-
  Penny is (Pa
  write("HalfD
  write("\nQua
  write("\nDim
  write("\nNic
  write("\nPen
%want to use a


/*calc_coins(C
  HalfDollar #=
  HalfDollar #=
  Quarter #=< 
  Quarter #=> (
  Dime #=< 3,
  Dime #=> 0,
  Nickel #=< 2,
  Nickel #=> 0,
```

```
?- change(1053, 1100, X).
HalfDollars: 0
Quarters: 1
Dimes: 2
Nickels: 0
Pennies: 2
X = [0, 1, 2, 0, 2]
Unknown action:  (h for help)
Action?
Unknown action: [ (h for help)
Action?
Unknown action: A (h for help)
Action? .

?- change(1050, 1100, [1,0,0,0,0]).
HalfDollars: 1
Quarters: 0
Dimes: 0
Nickels: 0
Pennies: 0
true .

?- 
```