Sprawozdanie z JIPP 2, 13.06.2020
Piotr Szwarc, 140789

Zad. 10

```
-module(zad10).
-export([test/0, iunz/1,iunz/2, runz/1]).
%Iteracyjnie
iunz([])->[];
iunz(X)-> iunz(X,[]).
iunz([],X)-> X;
iunz([H|T],X) ->
if H<0 -> iunz(T,lists:append(X,[0]));
true -> iunz(T,lists:append(X,[H]))
end.

%Rekurencyjnie
runz([])->[];
runz([H|T]) ->
if H<0 ->
lists:append([0],runz(T));
true ->
lists:append([H],runz(T))
end.

test() ->
X = [-1,-2,3,4,-5],
io:fwrite("Lista Wejsciowa:~w~n",[X]),
Y=iunz(X),
io:fwrite("Iteracja:~w~n",[Y]),
Z=runz(X),
io:fwrite("Rekurencja:~w~n",[Z]).
```

Zad. 11.1

```
-module(zad11).
-export([insertion_sort/1,select_sort/1, select_sort/2,join/2]).

insertion_sort(L) -> lists:foldl(fun insert/2, [], L).
insert(X,[]) -> [X];
insert(X,L=[H|_]) when X =< H -> [X|L];
insert(X,[H|T]) -> [H|insert(X, T)].

select_sort([]) -> [];
select_sort([X | []]) -> [X];
select_sort(ToSort) -> select_sort(ToSort, []).
select_sort([], Sorted) -> Sorted;
select_sort(Unsorted, Sorted) -> Max = lists:max(Unsorted),
select_sort(lists:delete(Max, Unsorted),
[Max] ++ Sorted).
```

```
join([],[])->[];
join(A,[])->A;
join([],B)->B;
join([H1|T1],[H2|T2])->
if H1<H2 ->
lists:append([H1],join(T1,lists:append([H2],T2)));
true ->
lists:append([H2],join(T2,lists:append([H1],T1)))
end.
```

Zad. 11.2

```
-module(zad112).
-export ([test/0, mapa/1, mapa/2]).

mapa(A)->
Map = maps:new(),
mapa(A,Map).

mapa([],Map)->Map;

mapa([H|T],Map)->
X = maps:get(H,Map,0),
mapa(T,maps:put(H,X+1,Map)).

test() ->
String = ['a','n','a','c','o','n',' d ','a'],
io:fwrite( "Ciąg: ~w~n " ,[String]),
Z=mapa(String),
io:fwrite( "Podsumowanie: ~w~n " ,[Z]).
```

Zad. 12

```
-module(zad12).
-export([akceptor/1,accept1/1,accept2/1,accept3/1,accept4/1]).

akceptor ([]) -> false ;
akceptor (A) -> accept1(A).
accept1 ([ 1 |T])->accept2(T);
accept1 ([ 0 |_] ) -> false .

accept2 ([ 0 , 1 |T]) -> accept2(T) ;
accept2 ([ 1 , 0 |T]) -> accept2(T) ;
accept2 ([ 0 , 0 , 0 |T]) ->accept3(T);
accept2 ([ 1 , 1 |_]) -> false .

accept3 ([ 1 , 1 |T] ) -> accept4(T) ;
accept3 ([ 1 , 0 |T]) -> accept4(T) ;
accept3 ([ 0 , 1 |_]) -> false .

accept4 ([ 0 , 0 ]) -> true ;
accept4 (L) -> accept3(L).
```