Задание 5
Султанов Шакир КА 4 курс
Методы сортировки списков в **ПРОЛОГЕ**
Задание состоит из 7-и частей. Для получения "5+" необходимо решить все 7 заданий. Некоторые из них содержат фрагменты кода, которые можно использовать при написании Ваших программ. Можно оформить одной программой, можно для каждого задания написать свою программу.

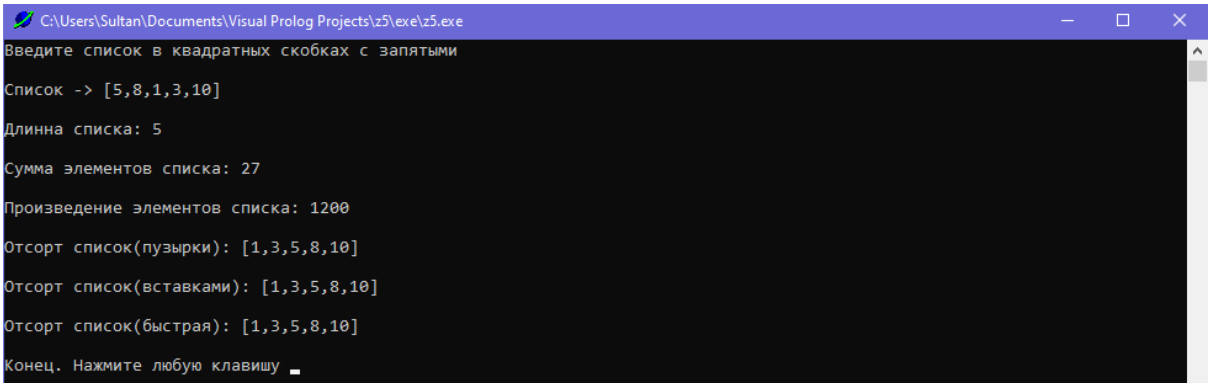
Половина заданий выполнено на Visual Prolog, а другая половина на SWI PROLOG
КОД:
Visual Prolog
1) Метод "пузырька"
2) Метод вставки.
3) "Быстрая" сортировка.

```
implement main
    open core, console

domains
    list = integer*.

class predicates
    len : (integer*, integer [out]).
    sum : (integer*, integer [out]).
    prod : (integer*, integer [out]).
    sort : (integer*, integer* [out]).
    swap : (integer*, integer* [out]) nondeterm.
    ins_sort : (integer*, integer* [out]).
    insert : (integer, integer*, integer* [out]).
    asc_order : (integer, integer) determ.
    quicksort : (list, list [out]).
    split : (integer, list, list [out], list [out]).
    concatenate : (list, list, list [out]).

clauses
    len([], 0) :-
        !.
    len([_X | Xs], 1 + S) :-
        len(Xs, S).

    sum([], 0) :-
        !.
    sum([X | Xs], S + X) :-
        sum(Xs, S).

    prod([], 1) :-
        !.
    prod([X | Xs], P * X) :-
        prod(Xs, P).
```

```
    sort(L, S) :-
        swap(L, M),
        !,
        sort(M, S).
    sort(L, L) :-
        !.
    swap([X, Y | R], [Y, X | R]) :-
        X > Y.
    swap([X | R], [X | R1]) :-
        swap(R, R1).

    ins_sort([], []).
    ins_sort([X | T1], L) :-
        sort(T1, L1),
        insert(X, L1, L).
    insert(X, [Y | L], [Y | L1]) :-
        asc_order(X, Y),
        !,
        insert(X, L, L1).
        /* вставляет первый аргумент в список, заданный в качестве второго аргумента,
результат третий аргумент */
    insert(X, L, [X | L]).
        /* вспомогательный предикат */
    asc_order(X, Y) :-
        X > Y.

    quicksort([], []).
    quicksort([Head | Tail], SortedList) :-
        split(Head, Tail, SList, BList),
        quicksort(SList, SList1),
        quicksort(BList, BList1),
        concatenate(SList1, [Head | Blist1], SortedList).

    split(_, [], [], []).
    split(Item, [Head1 | Tail1], [Head1 | SList], BList) :-
        Item > Head1,
        !,
        split(Item, Tail1, SList, BList).
    split(Item, [Head1 | Tail1], SList, [Head1 | BList]) :-
        split(Item, Tail1, SList, BList).

    concatenate([], List, List).
    concatenate([Item | List1], List2, [Item | List3]) :-
        concatenate(List1, List2, List3).

    run() :-
        init(),
```

```
        write("Введите список в квадратных скобках с запятыми"),
        nl,
        nl,
        write("Список -> "),
        List1 = read(),
        nl, % Список чисел
        _ = readchar(),
        len(List1, A),
        write("Длинна списка: ", A),
        nl,
        nl,
        sum(List1, S),
        write("Сумма элементов списка: ", S),
        nl,
        nl,
        prod(List1, P),
        write("Произведение элементов списка: ", P),
        nl,
        nl,
        sort(List1, NewList),
        write("Отсорт список(пузырки): ", NewList),
        nl,
        nl,
        ins_sort(List1, ListVstavka),
        write("Отсорт список(вставками): ", ListVstavka),
        nl,
        nl,
        quicksort(List1, ListQuick),
        write("Отсорт список(быстрая): ", ListQuick),
        nl,
        nl,
        write("Конец. Нажмите любую клавишу "),
        _ = readchar().

end implement main

goal
    console::runUtf8(main::run).
```

Результат:



КОД:

SWI PROLOG

4) Сортировка выбором

5) Слияние отсортированных списков

6) Сортировка слияниями

7) Проверка списка на упорядоченность

```
merge(List, List, []).
merge(List, [], List).

merge([MinList1|RestMerged], [MinList1|RestList1], [MinList2|RestList2]) :-
  MinList1 =< MinList2,
  merge(RestMerged,RestList1,[MinList2|RestList2]).
merge([MinList2|RestMerged], [MinList1|RestList1], [MinList2|RestList2]) :-
  MinList2 =< MinList1,
  merge(RestMerged,[MinList1|RestList1],RestList2).
% --------------------------Соритров-------------------------------------C
% Сортировка слиянием

sliyaniyeSort([], []).
sliyaniyeSort([A], [A|[]]).

sliyaniyeSort(Sorted, List) :-
  length(List, N),
  FirstLength is //(N, 2),
  SecondLength is N - FirstLength,
  length(FirstUnsorted, FirstLength),
  length(SecondUnsorted, SecondLength),
  append(FirstUnsorted, SecondUnsorted, List),
  sliyaniyeSort(FirstSorted, FirstUnsorted),
  sliyaniyeSort(SecondSorted, SecondUnsorted),
  merge(Sorted, FirstSorted, SecondSorted).


%-----------------------------------------------------------------
%Соритровка два сортированных списков
```

```
%
sortLists([],L,L).
sortLists(L,[],L).
sortLists([Head1|Tail1], [Head2|Tail2], L) :-
   Head1 < Head2 -> L = [Head1|R], sortLists(Tail1,[Head2|Tail2],R) ;
   Head1 > Head2 -> L = [Head2|R], sortLists([Head1|Tail1],Tail2,R) ;
   L = [Head1,Head2|R], sortLists(Tail1,Tail2,R).
%---------------------------------------------------------------
%Соритровка выбором
%
viborSort([],[]).
viborSort([M1|S],[H|T]):-min(H,T,M1),remove(M1,[H|T],N),viborSort(S,N).

min(M,[],M).
min(M,[H|T],M1):-min2(M,H,N),min(N,T,M1).

min2(A,B,A):-less(A,B).
min2(A,B,B):-not(less(A,B)).

less(A,B):-(A<B).

append([],B,B).
append([H|A],B,[H|AB]):-append(A,B,AB).

remove(X,L,N):-append(A,[X|B],L),append(A,B,N).
%---------------------------------------------------------------
%Проверка на упорядоченность
%
ordered( []     ) .
ordered( [_]    ) .
ordered( [X,Y|Z] ) :- X =< Y , ordered( [Y|Z] ) .
```

Результат:

```
% c:/Users/Sultan/Documents/Prolog/2.pl compiled 0.00 sec, 0 clauses
?- viborSort(S,[5,3,2,7]).
S = [2, 3, 5, 7] .

?- sortLists([5,7,2],[-2,2,3],S).
S = [-2, 2, 3, 5, 7, 2] .

?- ordered([6,2,4]).
false.

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

?- sliyaniyeSor(S,[2,7,1,9,5]).
Correct to: "sliyaniyeSort(S,[2,7,1,9,5])"? yes
S = [1, 2, 5, 7, 9]
```