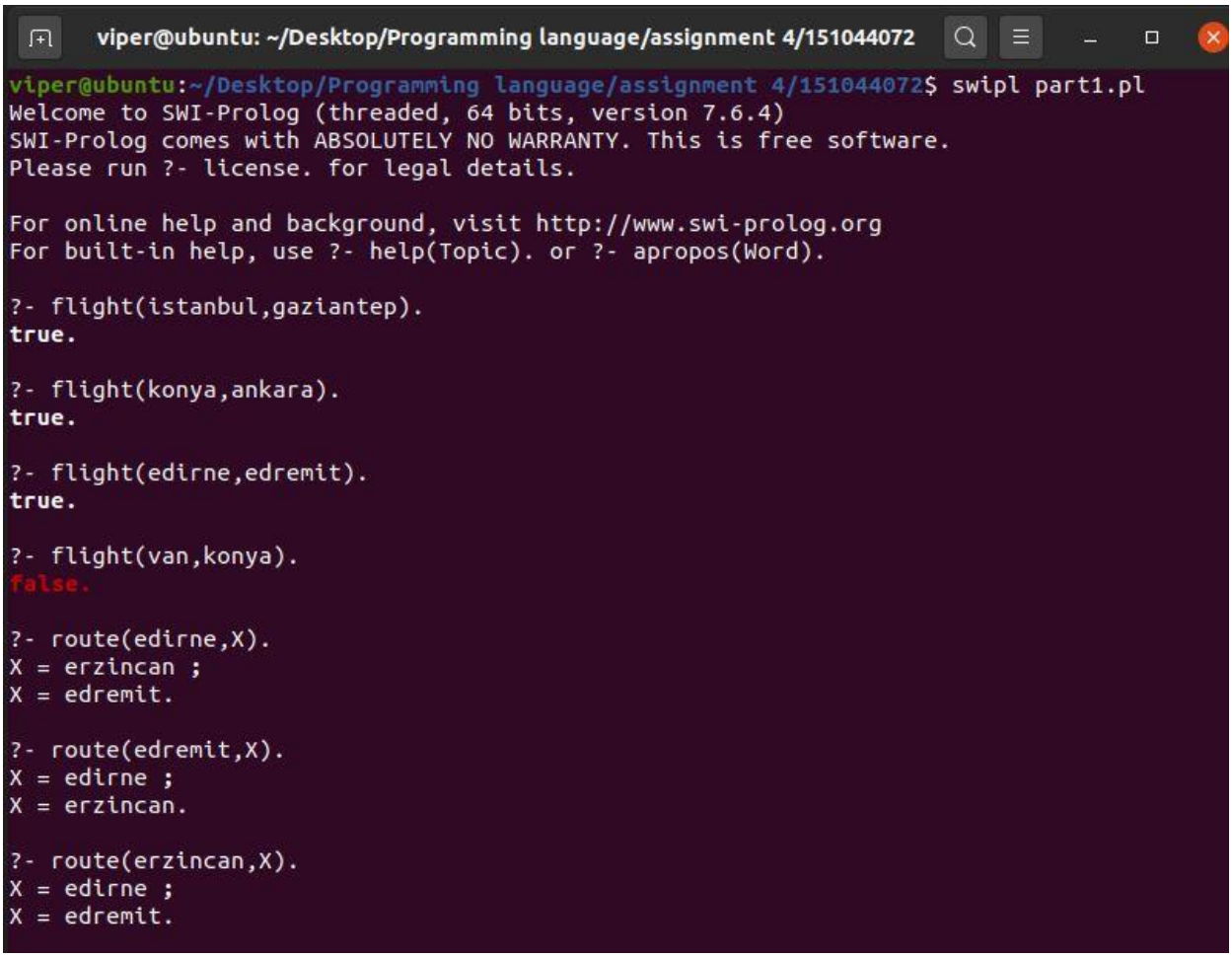
PROGRAMMING LANGUAGES - HW4 REPORT

ALPER YAŞAR – 151044072

Part 1 :

To check if you are in the same city, call the empty list and the route function with different parameters as first, last and first city again. Enter the city that arrives first in the blank list (K) and find the new destination city with flight information. Then search themselves with new city, last city, empty list, previous city. Find all the neighbors of the launched city and then the neighbor of the first city's neighbor and add it to the list of these possible cities. This search algorithm is Breath First Search (BFS). Finally, when the last city is the same as the previous city or the next city is the same as the city you want to go to, the search and add process ends.



Part 2:

As I said in part 1, check the route function with different parameters as an empty list and if you are in the same city as the first, last and again first city. Enter the city that arrives first in the blank list (V) and find the new destination city with flight information. Then search themselves with new city, last city, empty list, previous city. Find all the neighbors of the launched city and then the neighbor of the first city's neighbor and add it to the list of these possible cities. This search algorithm is Breath First Search (BFS). Finally, when the last city is the same as the previous city or the next city is the same as the city you want to go to, the search and add process ends. I added distance to part 1. Whenever a new neighbor was found I added their distance and finally found the minimum values.

```
viper@ubuntu:~/Desktop/Programming language/assignment 4/151044072$ swipl -s part2.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- sroute(ankara,konya,X).
227
X = 227 .

?- sroute(edremit,erzincan,X).
736
X = 736 .

?- sroute(edirne,erzincan,X).
[]
X = 1650 .

?- sroute(rize,van,X).
373
X = 373 .

?- sroute(rize,konya,X).
[antalya,istanbul,rize]
[rize]
[]
X = 1641 .

?- halt.
```

Part 3:

“schedule(S,P,T)”

First check the enrollment facts and facts will return classes for the student. Then check the class facts with class name, class facts will return times and places.

“usage(P,T)”

Call the class fact with place, you do not have to fill class name, class facts will return all usage times of a classroom(place).

“conflict(X,Y)”

This function checks, when you give an input like class, conflict on time and place. First take place results from class facts with given class name. After this take the time result from class facts with given class name. And last “or” the results.

“meet(X,Y)”

Find the classes that students take from class facts. At least “and” the results.

```
viper@ubuntu:~/Desktop/Programming language/assignment 4/151044072$ swipl -s part3.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- schedule(a,P,T).
P = 102,
T = 10 ;
P = 108,
T = 12.

?- schedule(b,P,T).
P = 102,
T = 10.

?- schedule(d,P,T).
P = 341,
T = 14.

?- usage(207,T).
T = 16 ;
T = 17.

?- usage(z23,T).
T = 10.

?- conflict(452,455).
true.

?- conflict(108,102).
false.

?- meet(a,b).
true .

?- meet(a,c).
true.

?- meet(d,e).
false.
```

Part 4

“element(E,S)”

Returns true if E(element) is in S(set). If first element of set is same with sending element returns true. The function has been designed to call "cdr sets" (_ | T) to check the first condition.

“union(S1,S2,S3)”

If S3 is the combination of S1 and S2, it returns true. First, it checks whether S1 or S2 or both are empty. If not null, it checks if the first element of S1 is the same as the element at S3, then checks S2. If so, cdr continues the same process for S1 and S2. If some are blank, the function will change and S3 is checking full set. Returns true if both are empty. However, if some items are not in S3, the loops are interrupted by an incorrect response.

“intersect(S1,S2,S3)”

Returns true if S3 is the intersection of S1 and S2. Checks first element of S3 is element in S1 and S2 or not. If it is, S1 and S2 delete this element and create new sets and calls the function that has parameter as new S1 and S2 sets and cdr Z. If S1 or S2 is null, returns as true, else if S3 is null first then element of S2 are in S1 lists or not, checks by one by. Even one element of S2 in S1 lists, returns false.

“equivalent(S1,S2)”

Returns true if S1 and S2 are equivalent sets. If size of S1 and S2 is same, return true. Calls the size function that its parameter is S1, and counts the size of S1 until S1 will be null. After S1, same process is doing for S2. And last if size of S1 and S2 is same, return true; else false.

```
viper@ubuntu:~/Desktop/Programming language/assignment 4/151044072$ swipl -s part4.pl
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- element (c,[a,b,c,d,e,f,2,4]).
ERROR: Syntax error: Operator expected
ERROR: element
ERROR: ** here **
ERROR:  (c,[a,b,c,d,e,f,2,4]) .
?- element(c,[a,b,c,d,e,f,2,4]).
true .

?- element(x,[a,b,c,d,e,f,2,4]).
false.

?- union([a,b,c,d],[1,2,3,4,5],[a,1,b,2,3,c,d,4,5]).
true .

?- intersect([a,3,4,7],[a,f,7,5,3,g],[a,7,3]).
true .

?- intersect([a,3,4,7],[a,f,7,5,3,g],[a,7]).
false.

?- equivalent([a,5],[y,0]).
true.

?- equivalent([a,5],[a]).
false.
```

Part 5

Read each line in “input.txt” in “getLines(Line)” function and separate numbers into list. Taking last element as result “last(NewList,Result)” and delete it from list in “list_delete(X, [Y|L2], [Y|L1])”. Taking length of new list for creating permutation for found appropriate operators (+,-,*) for calculation.

“count_for_make_operator(X,Len,Z,Ops,NumbersList,Result)” recursive function creating operators randomly.

When operator production is done, taking generate operators randomly, “take_ops(X,Len,OpList,NewList,NumbersList,Result)” and in “chooseRandom(OpList,Op)” .

When operators choosing finished pass to take number randomly “take_num(0,Length,OpList,NewList,NumbersList,[],Result)”. This function creating new array for calculation with result. When array created passing to calc function.

“calc(OpList,WriteOp,NewOpList,NewNumberList,NumbersList,Len,Calc,Result)” function taking numbers and operators from array and delete it and calculating each two numbers and adding result to array.

I write true result to output.txt but numbers placement is change. Because how operator found and numbers get exact result it return it and write it to file.