Adam Cherek s464922

zadanie algorytmiczne nr 13:

algorytm będzie dla języków L1 i L2, przechodził po kolejnych głowach języków, konkatenując każde słowo języka L1 z każdym słowem języka L2.

funkcja CONLANG(listaL1, listaL2): zwraca język będący konkatenacją języków L1 i L2 przy pomocy funkcji CON(P,Q) i MEMBERSHIP(L,P)

```
{
        list L1:= listaL1
        list L2:= listaL2                   % deklarujemy potrzebne zmienne %
        list Pom:= [-]

        if L1 = [-] then                    % sprawdzamy czy żaden język nie jest pusty %
                return L2
        if L2 = [-] then
                return L1

        if MEMBERSHIP(Pom, CON(HEAD(L1), HEAD(L2))) = NIE then
                MAKELIST(CON(HEAD(L1), HEAD(L2)), Pom)

        while TAIL(L1) != [-] do            % konkatenujemy wszystkie słowa z L1 ze słowami z L2 %
                L1:= TAIL(L1)
                L2 := listaL2
                if MEMBERSHIP(Pom, CON(HEAD(L1), HEAD(L2))) = NIE then
                        MAKELIST(CON(HEAD(L1), HEAD(L2)), Pom)
                while TAIL(L2) != [-] do
                        L2 := TAIL(L2)
                        if MEMBERSHIP(Pom, CON(HEAD(L1), HEAD(L2))) = NIE then
                                MAKELIST(CON(HEAD(L1), HEAD(L2)), Pom)

        return Pom

}
```

zadanie algorytmiczne nr 14:

wykorzystując to że potęgą języka jest jego n-krotna konkatenacja z samym sobą algorytm będzie n-krotnie konkatenował język L z kolejnymi potęgami języka L obliczonymi we wcześniejszym wywołaniu

```
funkcja POWLANG(listaL , liczbaN): zwraca język będący n-tą potęgą języka L
{
	list L:= listaL
	int Pow:= 0
	list Pom:= [[]]

	while Pow < liczbaN do
		Pom:= CONLANG(Pom, L)
		Pow:= Pow +1
	return Pom
}
```