Adam Cherek s464922

zadanie algorytmiczne nr 11:

algorytm będzie dla zadanej listy słów L przechodził po kolejnych głowach tej listy dodając pierwsze wystąpienia każdego słowa do listy pomocniczej i zwracając tę listę pomocniczą będącą listą słów L bez powtórzeń czyli językiem

funkcja REMOVEREPETITIONS(listaL): zwraca listę L bez powtórzeń przy pomocy funkcji MEMBERSHIP(L,P)

```
{
        list L:= listaL
        list Pom:= [-]
        if L = [-] then    % sprawdzamy możliwe sprzeczności %
                return Pom
        else                                % jeżeli nie ma sprzeczności to zaczynamy
                                              przechodzić po głowach listy L
                if MEMBERSHIP(Pom, HEAD(L)) = NIE then
                        MAKELIST(HEAD(L), Pom)
                while TAIL(L) != [-] do   %sprawdzamy kolejne słowa i ewentualnie dodajemy
                                            je do listy Pom%
                        L := TAIL(L)
                        if MEMBERSHIP(Pom, HEAD(L)) = NIE then
                                MAKELIST(HEAD(L), Pom)
                return Pom      %zwracamy język %
}
```

zadanie algorytmiczne nr 12:

algorytm będzie szukał kolejnych słowa z listy L1 w liście L2 i usuwał je z listy L2, jeżeli w momencie ukończenia przejścia po liście L1, lista L2 będzie pusta a wszystkie dotychczasowe słowa z L1 się w niej znajdowały to zwróci TAK, w przeciwnym wypadku zwróci NIE

funkcja EQLANG(listaL1 , listaL2): zwraca TAK lub NIE w zależności czy języki są sobie równe wykorzystując funkcje REMOVE(L,P) i MEMBERSHIP(L,P)

```
{
        list L1 := listaL1
        list L2 := listaL2                      % deklarujemy potrzebne zmienne%

        if L1 = [-]  and L2 != [-] then         % sprawdzamy możliwe sprzeczności %
                return NIE
        if L2 = [-] and L1 != [-] then
                return NIE
        if MEMBERSHIP(L2, HEAD(L1)) = NIE then          % sprawdzamy pierwszy element L1 %
                return NIE
        else
                L2 := REMOVE(L2,(HEAD(L1))

        while TAIL(L1) != [-] do                        % sprawdzamy pozostałe
                                                            elementy L1 %
                L1 := TAIL(L1)
                if MEMBERSHIP(L2, HEAD(L1)) = NIE then
                        return NIE
                else
                        L2 := REMOVE(L2,(HEAD(L1))

        if L2 = [-] then                                % jeżeli po sprawdzeniu
                                                        wszystkich elementów L1, L2
                                                        też jest już puste to zwracamy
                                                        TAK %
                return TAK
        else
                return NIE
}
```