```
type graph <firth : adrVertex>
type adrVertex : pointer to vertex
type adrEdge : pointer to edge
type vertex : < id : char,
                next : adrVertex,
                firstEdge : adrEdge>
type edge : <idVertex : char,
             weight   : integer,
             next     : adrEdge>
```

1. Function inDegree(G:graph, v: adrVertex) -> integer

Kamus

```
P : adrVertex
Q : adrEdge
total : integer
```

Algoritma

```
P = G.first
total = 0
if P != NULL then
  while P != NULL do
    Q = P -> firstEdge
    while (Q != NULL) do
```

```
        if Q->idVertex == v-> id then
          total = total +1
        endif
        Q = Q->next
      endwhile
      p = p->next
    endwhile
    return total
  endif
  return -1
endfunction
Function outDegree(G:graph, v: adrVertex)->
integer
Kamus
    Q : adr Edge
    total : integer
Algoritma
  if v != NULL then
    Q = v -> firstEdge
    total =0
    while Q != NULL do
```

```
        total = total+1
        Q = Q -> next
    endwhile
    return total
  endif
  return -1
endfunction
```

3 Function degree(G:graph,v:adrVertex)->integer

Kamus

Function inDegree(graph,adrVertex)->integer

Function outDegree(graph,adrVertex)->integer

Algoritma

return inDegree(G,v)+outDegree(G,v)

endfunction

4 Function isSimpleGraph(G:graph)->boolean

Kamus

P : adrVertex

Q : adrEdge

Algoritma

if G.first != NULL then

```
P = G.first
while P != NULL do
    Q = P->firstEdge
    while Q != NULL do
        if P->id == Q->idVertex then
            return false
        endif
        Q = Q->next
    endwhile
    P = P->next
endwhile
endif
return false
endfunction
```