Problem Statement:

A student wants to create his dictionary to note the words and its definitions. Via provided menu, student can **add**, **find**, **delete**, **print** a word. Student can also see the all words in the **multimap**.

Justification:

For this project, the best container for this kind of dictionary is **multimap**.
Because, word's name is the **unique key** and one word can have multiple definitions.

Word:

name : string
definition : string

Word Operations:

init(**name, definition**)
AddName()
GetName()
SetName(**name**)
PrintName()

AddDefinition()
GetDefinition()
SetDefinition(**definition**)
PrintDefinition()

PrintWord()
PrintAllProperties()

operator == ()
operator != ()

Node:

INFO : TElem
NEXT : Node

Node Operations:

init (**info**)
info ()
setInfo (**info**)
next ()
setNext (**next**)
operator == ()

SinglyLinkedList:

SIZE : Integer
MAX_SIZE : Integer
HEAD : ↑Node
TAIL : ↑Node

SinglyLinkedList Operations:

init ( )
size ()
max_size ()
head ()
pop_front ()
pop_back ()
empty ()
full ()
push_front (**givenElement**)
push_back (**givenElement**)
push_at (**givenElement, position**)
push_after (**givenElement, specifiedElement**)
find ()
findPreviousFrom (**givenElement**)
remove_front ()
remove_back ()
remove (**givenElement**)

SSLIterator:

SLL : ↑SinglyLinkedList
CURRENT : ↑Node
POSITION : Integer

SSLIterator Operations:

init (**sll**)
current ()
valid ()
position ()
next ()
operator ++ ()

Pair:

KEY : TKey
VALUE : TValue

Pair Operations:

init (**key, value**)
key ()
value ()
setValue (**value**)
operator == ()
operator != ()

HashTable:

LENGTH : Integer
lists : SinglyLinkedList [ ]
HashFunction (**itemKey**)

HashTable Operations:

init ()
put (**Pair**)
get (**Pair**)
length ()

MultiMap:

containerSize : Integer
table : HashTable

MultiMap Operations:

init (**mm**)
{
post: **mm** ∈ MultiMap, **mm** = ∅
}

destroy (**mm**)
{
pre: **mm** ∈ MultiMap
post: **mm** was destroyed (allocated memory was freed)
}

insert (**key**, **value**)
{
pre: key ∈ Tkey, value ∈ Tvalue
post: the pair <k,v> was added into **mm**. If the pair <k,v> is already in the **mm**, nothing is added.
}

find (**key**)
{
pre: key ∈ Tkey

post: true → returns a list of values associated with key.

false → returns an empty list.
}

```
remove ( key, value )
{
pre:    key ∈ Tkey, value ∈ Tvalue
post:   the pair <k,v> was deleted from mm.
}

size ()
{
post:   returns the containerSize.
}

empty ()
{
post:   true    →   if mm is not empty
        false   →   otherwise
}

clear ()
{
post:   removes all pairs in the mm.
}

count ( key )
{
pre:    key ∈ Tkey
post:   returns the number of pairs whose keys
        are equal to key.
}
```

MMAPIterator:

map : MultiMap

MMAPIterator Operations:

```
init ( map )
{
pre:    map ∈ MultiMap
post:   it is an iterator over map
}

current ()
{
pre:    iterator is valid
post:   returns current element
}

valid ()
{
post:   true    →   current element is not NULL
        false   →   otherwise
}

next ()
{
pre:    iterator is valid
post:   iterates to the next element
}

operator ++()
{
pre:    iterator is valid
post:   iterates to the next element
}
```