LOOKUP TABLE

KEYWORDS	
tamsayi	This is the keyword for defining an integer variable. If this keyword is used, then next token should be an identifier *(for example: tamsayi numara1;)*. After the identifier there should be **' ; '** sign to end this definition Also if you want to define an integer array variable you should use **' ['** sign after the identifier and then there should be an integer variable or integer number. After this there should be **'] '** sign. Integer numbers' values will be 1 by default.
eger	This keyword is used to identify **' if '** structure. After the keyword there should be two brackets and between these brackets, there should be any condition *(for example: 5 != 6 , 6 > numara1 , arr1[8] = arr2[numara1])*. If this condition is true, then after these there should be some expressions between *basla* keyword and *bitir* keyword. These expressions can be assignments, multiplications, definitions etc.
iken	This keyword is used to identify while structure. After the keyword there should be brackets. Between the brackets you need to enter a condition like in if structure. After the condition part there should be basla keyword and the you can write the series of statements. You should end the structure with the bitir keyword.
karakter	This is the keyword for defining a character variable . After this keyword there should be an identifier *(for example: karakter harf1;)*. After the identifier if there is **' ['** sign that means this variable is going to be and character array. After this sign there should be an integer variable or numbers. Also, you have to close the bracket with **'] '** sign. Also, there should be **' ; '** sign at the end of this definition. *karakter* variables also behaves like strings when concatenation process is done. Character variables' values will be ' q ' by default.
ondalik	This is the keyword for defining a float variable. If this keyword is used, then next token should be an identifier *(for example: ondalik numara2;)*. After the identifier there should be **' ; '** sign to end this definition.
topla	This is the keyword for addition operation for integers. It simply does the duty of the **' + '** sign. After the *topla* keyword there should be two brackets and between them there should be two variable or numbers. Also *topla function* is a recursive function so it can be called inside the *topla function (for example: topla(5,topla(number1,arr1[number2])))*. Also, this function returns to an integer value and it does not make assignments. To make assignment it should be used inside the assignment function which has the keyword **"ata"**.
carp	This is the keyword for multiplication operation for integers. It simply does the duty of the **' * '** sign. After the *carp* keyword there should be two brackets and between them there should be two variable or numbers. Also *carp function* is a recursive function so it can be called inside the *carp function (for example: carp(5,carp(number1,arr1[number2])))*. Also, this function returns to an integer value and it does not make assignments. To make assignment it should be used inside the assignment function which has the keyword **"ata"**.
	This is the keyword for subtraction operation for integers. It simply does the duty of the **' - '** sign. After the *cikar* keyword there should be two brackets and between them there should be two variable or numbers. Also, *cikar*

cikar	*function* is a recursive function so it can be called inside the *cikar function (for example: cikar(5,cikar(number1,arr1[number2])))*. Also, this function returns to an integer value and it does not make assignments. To make assignment it should be used inside the assignment function which has the keyword **"ata"**.
bol	This is the keyword for division operation for integers. It simply does the duty of the **' / '** sign. After the *bol* keyword there should be two brackets and between them there should be two variable or numbers. Also, *bol function* is a recursive function so it can be called inside the *bol function (for example: bol(5,bol(number1,arr1[number2])))*. Also, this function returns to an integer value and it does not make assignments. To make assignment it should be used inside the assignment function which has the keyword **"ata"**.
toplaond	This is the keyword for addition operation for float numbers. After this keyword there should be brackets. Inside these brackets there should be two float numbers or variables *(there should be comma between two variables or numbers)*. *toplaond* function is a recursive function so it can be called in itself. Also, this function returns to a float value. Basically, same of the topla function but for float numbers.
cikarond	Same of the function toplaond but this function is <u>for subtraction</u>.
carpond	Same of the function toplaond but this function is <u>for multiplication</u>.
bolond	Same of the function toplaond but this function is <u>for division</u>.
dogruluk	The dogruluk data type is used to store only two possible values: dogru and yanlis. This keyword is for defining Boolean variables. After this keyword there should be an identifier *(for example: dogruluk sorgula;)*. Also, there should be **' ; '** sign at the end of the definition.
dogru	This keyword is for Boolean condition <u>true</u>. it corresponds to true keyword in Java. It can be used in 'if and while' statements condition part *(for example: eger(dogru))*. If *dogru* keyword is used, then it runs the statements inside of if structure.
yanlis	This keyword is for Boolean condition <u>false</u>. it corresponds to false keyword in Java. It can be used in 'if and while' statements condition part *(for example: eger(yanlis))*. If *yanlis* keyword is used, then it does not allow to run the statements inside of if structure.
basla	This keyword is used inside 'if and while' structures. It simply shows that the loop or selection structure began.
bitir	This keyword is used inside of 'if and while' structures. It shows that statements end.
ata	This keyword is used to make assignment operations. After that, the first variable inside of the brackets must be already defined variable *(array, integer, float, character)*. The second variable inside of the brackets can be 'array, integer, float, character, variable or number'. To make the assignment both variables or numbers must be in same type *(for example: Assume that, number1 , arr1[n] is already defined integer variables before the assignment function, and assignment is done like: ata(number1,5) or ata(number1,arr1[number1]))*.But catenation operation is an exception because if there is a catenation operation then after the assignment operation that variable behaves like it is a string. If you assign just one character to that string, then it becomes a character again and behaves like a character. Catenation process is done like ata *(karakter1, m+e+r+h+a+b+a)*.

	Now karakter1 behaves like it is a string. If you make the assignment operation like this: ata(karakter1, merhaba) then the value of karakter1 will be which is the first letter of the merhaba identifier.

SIGNS

>	*The greater sign* is used in 'if and while' structures condition part. It is used to check the first condition is greater than the second condition.
<	This *smaller sign* is used in 'if and while' structures condition part. It is used to check the first condition is smaller than the second condition.
=	Equal sign is used in 'if and while' structures condition part. It is used to check the first condition is equal to the second condition.
!	Exclamation mark is used in if and file structures condition part. It is used before the equal sign and it checks the first condition is not equal to the second condition.
;	This symbol is used after any integer, character or float definitions. It shows us that the definition part is over.
[	This symbol is used after definition part and before **';'** symbol. It shows us that an array is going to be defined. After this sign there must be an integer variable or number.
]	This symbol is used after definition part before **';'** symbol. It shows us that array definition ended and now there should be **';'** sign.
(	The left bracket means this is the beginning of the condition part in if, while or any arithmetic operations.
)	The right bracket means, that the condition part ends.
+	The plus sign is used for concatenation operation in assignment function *(for example: ata(harf1,m+e+r+h+a+b+a))*. After the concatenation operation the character behaves like a string. Also, this sign **is not** used in any arithmetic calculations.
-	The minus sign represents negative numbers. It is used in assignment function and arithmetic operations.

.	This sign is used to show floating point numbers.
,	It is used to separate two different variables or numbers.