HTML TAGS: <img> and <a>

COMPILER DESIGN

**Submitted To :-**

**Dr. Ankit Rajpal**

**Submitted by :-**

**Divya Solanki (18)**

**Vandana Yadav (61)**

## 2022

*MCA Sem IV*

## Department of Computer Science University of Delhi

**INDEX**

[Problem Statement 3](#_TOC_250004)

Syntax of <img> 3

Syntax of <a> 4

Assumption 5

[Test Cases(Valid) 4](#_TOC_250003)

[Test Cases(Invalid) 4](#_TOC_250002)

[Lex Source Code 5](#_TOC_250001)

[YACC Source Code 7](#_TOC_250000)

How to Compile 9

Outputs 11

References 12

# Problem Statement

To create a parser using lex and yacc to parse the HTML tags <img> and <a>

*Syntax of HTML Tags*

**<img> Tag**

**Syntax:**

**<****img src="image\_name.jpg" alt="text" >**

The <img> tag is used to embed an image in an HTML page.

The <img>  tag has two required attributes:

* src - Specifies the path to the image
* alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed

Attributes:

* [src](https://www.geeksforgeeks.org/html-img-src-attribute/)
* [alt](https://www.geeksforgeeks.org/html-img-alt-attribute/)
* crossorigin
* [height](https://www.geeksforgeeks.org/html-img-height-attribute/)
* [width](https://www.geeksforgeeks.org/html-img-width-attribute/)
* [ismap](https://www.geeksforgeeks.org/html-img-ismap-attribute/)
* loading
* longdesc
* referrerpolicy
* sizes
* srcset
* [usemap](https://www.geeksforgeeks.org/html-img-usemap-attribute/)

**<a> Tag**

**Syntax:**

**<a href="https://www.xyzz.com">Visit xyzz.com</a>**

### The <a> tag defines a hyperlink, which is used to link from one page to another.

### If the <a> tag has no href attribute, it is only a placeholder for a hyperlink.

### Attributes:

* [**charset**](https://www.geeksforgeeks.org/html-a-charset-attribute/)
* [**download**](https://www.geeksforgeeks.org/html-a-download-attribute/#:~:text=The%20download%20attribute%20is%20used,name%20of%20the%20downloaded%20file.)
* [**hreflang**](https://www.geeksforgeeks.org/html-a-hreflang-attribute/)
* [**media**](https://www.geeksforgeeks.org/html-a-media-attribute/)
* [**name**](https://www.geeksforgeeks.org/html-a-name-attribute/)
* [**rel**](https://www.geeksforgeeks.org/html-a-rel-attribute/)
* [**shape**](https://www.geeksforgeeks.org/html-a-shape-attribute/)
* [**type**](https://www.geeksforgeeks.org/html-a-type-attribute/)
* [**target**](https://www.geeksforgeeks.org/html-a-target-attribute/)
* [**rev**](https://www.geeksforgeeks.org/html-a-rev-attribute/)
* **ping**

# Assumptions

1. Assuming attributes are in lower case only for simplicity.
2. Compulsory attribute should come before non-compulsory attribute.
3. The case of nested tag is covered here.
4. Attribute values must be in either double quotes or single quotes.
5. Only <a> tag can have nested <a> or <img> tags.

**Test Cases for <img> tag (Valid)**

1. A img tag used with all compulsory attributes and other attributes.

Eg:

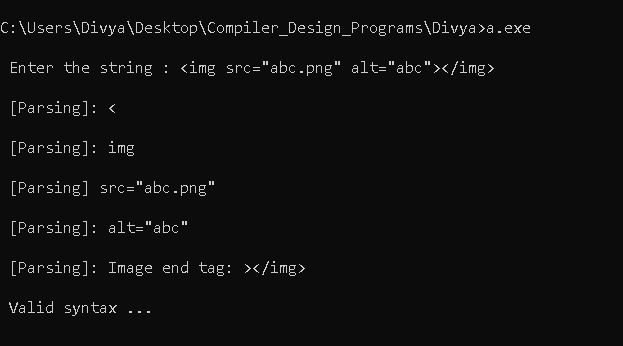
<img src=”abc.png” alt=”abc” height=”100” width=”100”>



1. A img tag with only compulsory attribute.

Eg:

<img src=”abc.png” alt=”abc”></img>



1. A img tag can end in three ways
   1. End with “></img>” [No whitespaes in between]

Eg. <img src=”abc.png” alt=”abc” height=”100” ismap=”1”> </img>



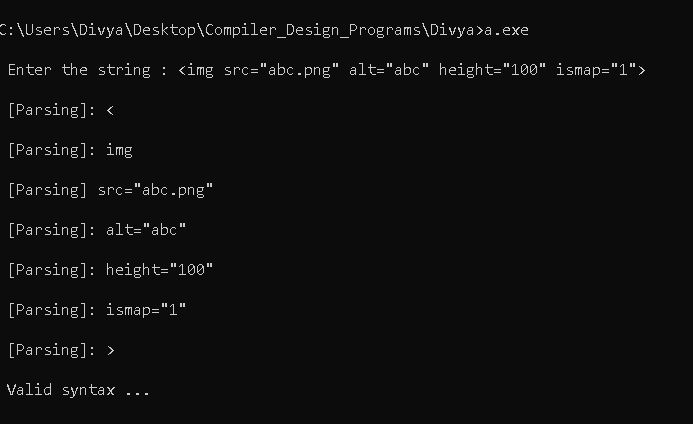
* 1. End with “/>”

Eg. <img src=”abc.png” alt=”abc” height=”100” ismap=”1”/>



* 1. End with “>”

Eg. <img src=”abc.png” alt=”abc” height=”100” ismap=”1”>



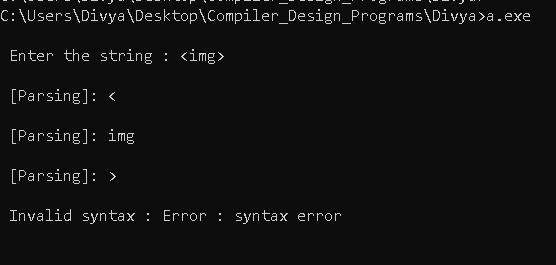
#### 

#### Test Cases for <img> tag (Invalid)

1. A img tag with no attribute.

Eg:

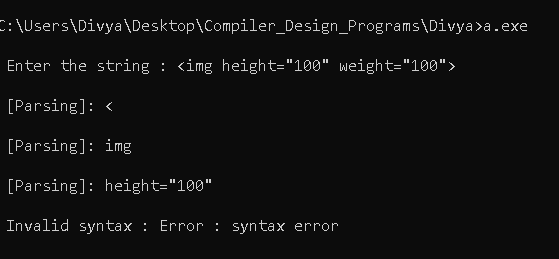
<img>



1. A img tag with compulsory attribute missing

Eg:

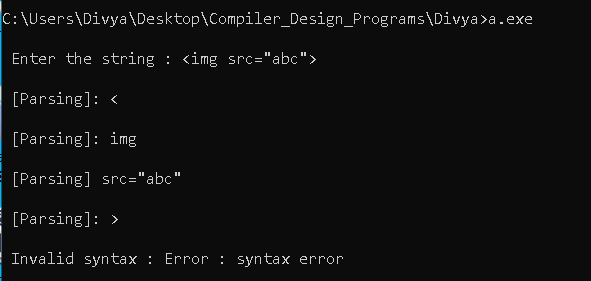
<img height=”100” weight=”100”>



1. A img tag with one compulsory attribute missing

Eg:

<img src=”abc”>



1. A img tag with all compulsory or optional attributes but invalid ending

Eg

* 1. End tag is not matched with starting tag

<img src=”abc.png” alt=”abc” height=”100” width=”100”>

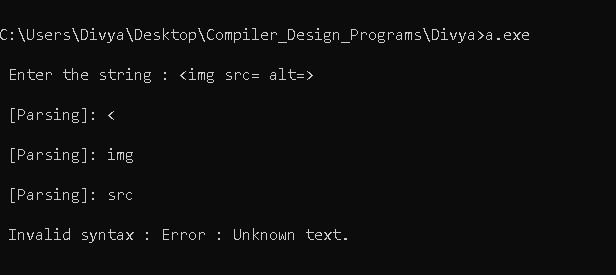
</b>



1. A img tag with missing attribute values

Eg:

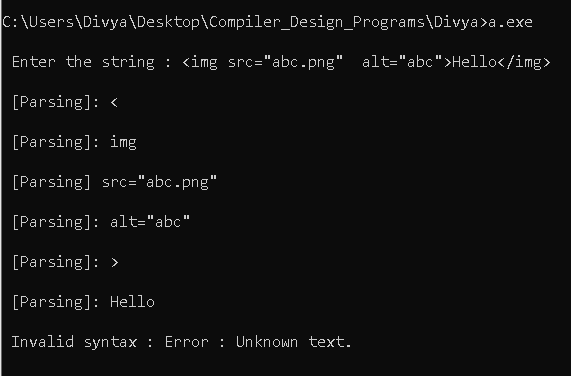
<img src= alt=>



1. A img tag with content inside tags

Eg:

<img src=”abc.png” alt=”abc”>Hello</img>

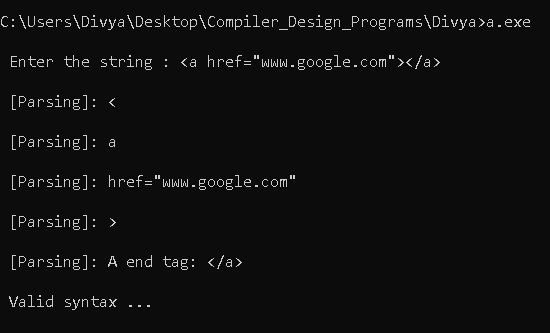


**Test Cases for <a> tag (Valid)**

1. A <a>tag used with attributes and same start and end tag.

Eg:

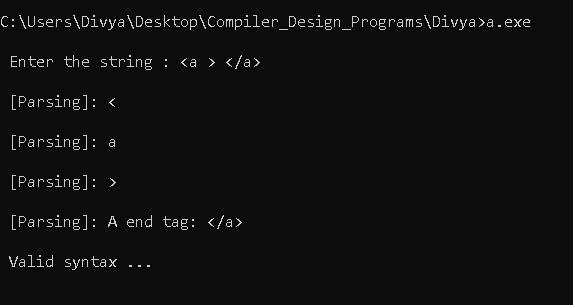
<a href=”www.google.com”> </a>



1. A <a>tag with no attribute

Eg:

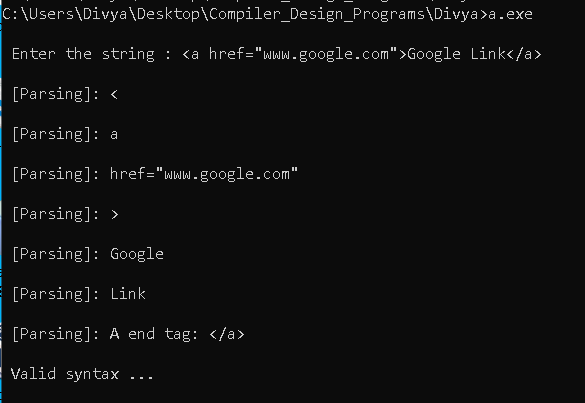
<a > </a>



1. A <a> tag can have content simple in it

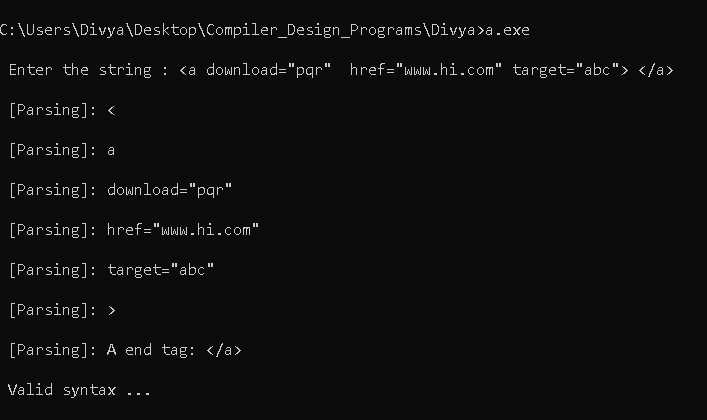
Eg:

<a href=”www.google.com”>Google</a>



1. A <a> tag with different attribute values in any order.

Eg: <a download=”pqr” href=”www.hi.com” target=”abc”> </a>

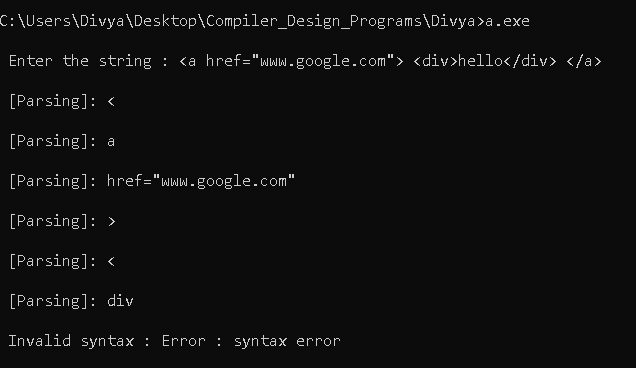


#### Test Cases for <a> tag (Invalid)

1. A <a> tag enclosing other tags

Eg:

<a href=”www.google.com”> <div>hello</div> </a>

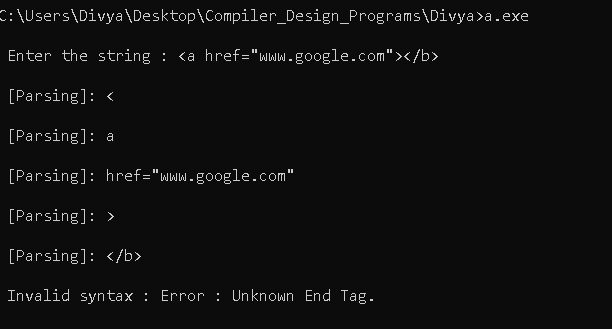


1. A <a> tag with or without attributes but invalid ending

Eg

* 1. End tag is not mached with starting tag

Eg: <a href=”www.google.com”></b>



* 1. Missing endtag

Eg: <a>

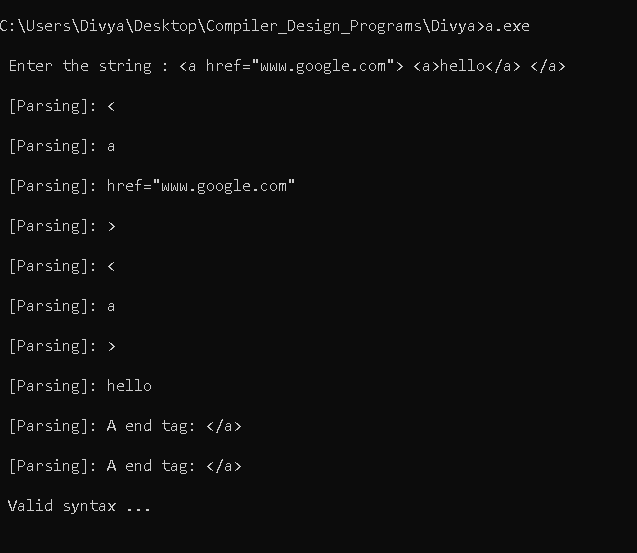


#### Test Cases for Nested Tags (Valid)

1. A nested tag enclosing <a> tags

Eg:

<a href=”www.google.com”> <a>hello</a> </a>



1. A nested tag enclosing <img> tags

Eg:

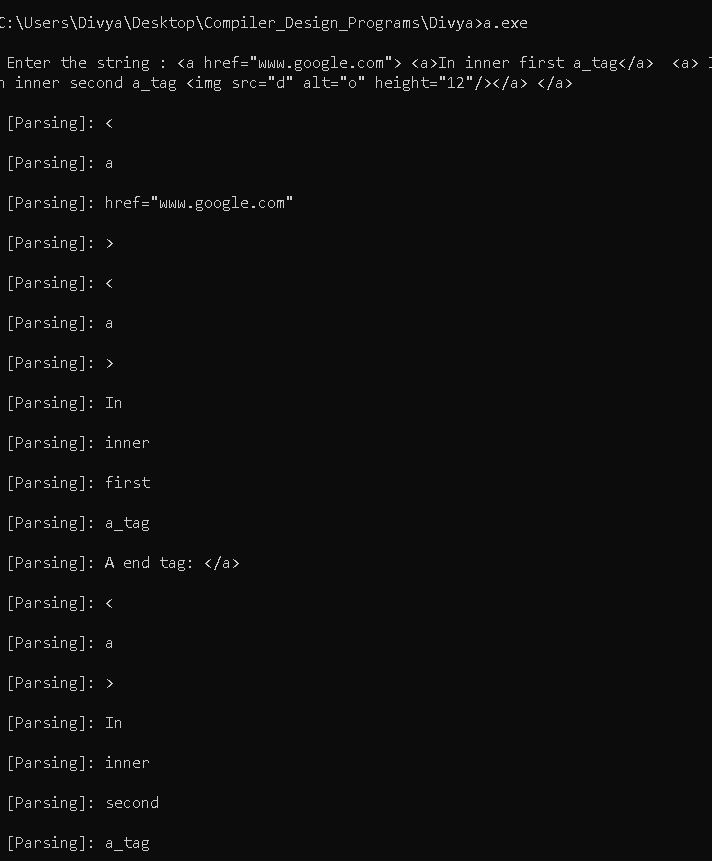
<a href=”www.google.com”> <img src=”d” alt=”o” height=”12”/> </a>

#### 

1. A nested tag enclosing both <a> and <img> tags in any order.

Eg:

<a href=”www.google.com”> <a>In inner first a\_tag</a> <a> In inner second a\_tag <img src=”d” alt=”o” height=”12”/></a> </a>



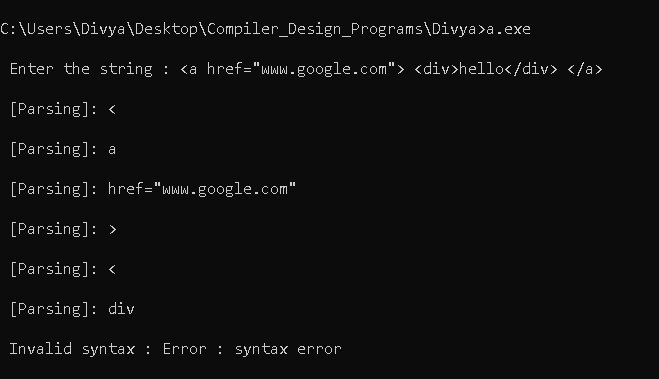
#### 

#### Test Cases for Nested Tags (Invalid)

* 1. A nested tag enclosing other tags

Eg:

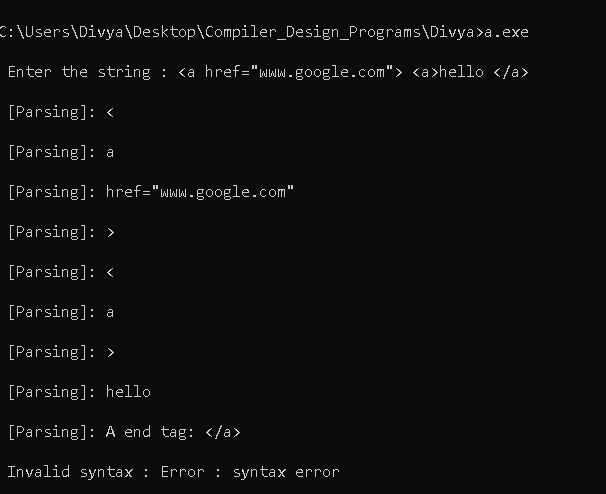
<a href=”www.google.com”> <div>hello</div> </a>



* 1. A nested tag enclosing other <a> or <img> tags without ending.

Eg:

<a href=”www.google.com”> <a>hello </a>



**LEX Code**

/\* Lex Program for { a^n c b^n } Language. \*/

%{

/\*contains declaration of all the tokens in th yacc program\*/

#include "y.tab.h"

/\*variables used while validation\*/

int a\_valid = 0,end\_tag1=0;

%}

/\*Regular expressions to be matched while performaing lexical analysis\*/

/\*RE for whitespaces\*/

Ws ([ \t ]+)

/\*RE for letters\*/

Letter [A-Za-z]

/\*RE for digits\*/

Digit [0-9]

/\*RE for identifiers\*/

Name {Letter}({Letter}|{Digit}|[\_])\*

/\*RE for content inside <a> tag\*/

Content {Name}|{Letter}|{Digit}

/\*attributes of <a> tag\*/

AAttrList "download"|"href"|"hreflag"|"media"|"ping"|"referrerpolicy"|"rel"|"target"|"type"

/\*attribute list for <a> tag\*/

AAttr {AAttrList}

/\*attributes of <img> tag\*/

ImgAttList "height"|"width"|"crossorigin"|"use-credentials"|"ismap"|"loading"|"longdesc"|"referrerpolicy"|"no-referrer-when-downgrade"|"origin"|"origin-when-cross-origin"|"unsafe-url"|"sizes"|"usemap"

/\*attribute list for image tag\*/

ImgOtherAtt {ImgAttList}

/\*RE for attribute value\*/

AttValue (\"[^<&"]\*\")|(\'[^<&']\*\')

//end of declaration section

%%

/\*translation rules

pattern {

action

}

\*/

/\* "<"" will be matched here.\*/

[<] { printf("\n [Parsing]: %s\n", yytext);

return START\_ANGLE\_BRACKET;}

/\* "a" will be matched here.\*/

[a] { /\*a\_valid is used for validating content of <a> tag.\*/

a\_valid =1;

printf("\n [Parsing]: %s\n", yytext);

return A\_TAG;}

"img" { printf("\n [Parsing]: %s\n", yytext);

return IMG\_TAG;}

{Ws} {;}

/\*anchor tag's attribute is matched here\*/

{AAttrList}{Ws}?[=]{Ws}?{AttValue}{Ws}? {

//valid attribute for a tag

printf("\n [Parsing]: %s\n", yytext);

return A\_ATTRIBUTE;

}

[>] { /\*end\_tag1 is used for validating content of <a> tag.\*/

end\_tag1=1;

printf("\n [Parsing]: %s\n", yytext);

return END\_TAG1;}

/\*End tag for </a> is matched here\*/

[<]{Ws}?[/]{Ws}?[a]{Ws}?[>] {

//valid endtag

printf("\n [Parsing]: A end tag: %s\n", yytext);

return A\_END\_TAG;

}

/\*Unlnown End tags matched here\*/

[<]{Ws}?[/]{Ws}?{Name}{Ws}?[>] {

//valid endtag

printf("\n [Parsing]: %s\n", yytext);

yyerror("Unknown End Tag.");

}

/\*content inside <a> tag is matched here.\*/

{Content} {

if(a\_valid==1 && end\_tag1==1)

{

printf("\n [Parsing]: %s\n", yytext);

return A\_CONTENT;

}

else

{

printf("\n [Parsing]: %s\n", yytext);

yyerror("Unknown text.");

}

}

/\*attribute src of image tag with attribute value is matched here.\*/

"src"{Ws}?[=]{Ws}?{AttValue}{Ws}? {

printf("\n [Parsing] %s\n", yytext);

return IMG\_SRC;

}

/\*attribute alt of image tag with attribute value is matched here.\*/

"alt"{Ws}?[=]{Ws}?{AttValue}{Ws}? {

printf("\n [Parsing]: %s\n", yytext);

return IMG\_ALT;

}

/\*all other valid img attributes is matched here\*/

{ImgOtherAtt}{Ws}?[=]{Ws}?{AttValue}{Ws}? {

//valid attribute for img tag

printf("\n [Parsing]: %s\n", yytext);

return IMG\_OTHER\_ATTRIBUTE;

}

/\*all other attributes which are invalid are matched here\*/

{Name}{Ws}?[=]{Ws}?{AttValue}{Ws}? {

//attribute list doesnot content this attribute

printf("\n [Parsing]: %s\n", yytext);

yyerror("Unknown attribute.");

}

/\*End tag for image "/>" is matched here\*/

[/]{Ws}?[>]{Ws}? {

//valid endtag for img

printf("\n [Parsing]: %s\n", yytext);

return IMG\_END\_TAG2;

}

/\*End tag for image "></img> is matched here\*/

[>]{Ws}?[<]{Ws}?[/]{Ws}?("img"){Ws}?[>]{Ws}? {

//valid endtag

printf("\n [Parsing]: Image end tag: %s\n", yytext);

return IMG\_END\_TAG3;

}

/\*newline character\*/

[\n] { return NL; }

/\*Everything else is matched here.\*/

. { printf("\n [Parsing]: %s\n", yytext);

yyerror("Unknown text.");}

%%

int yywrap()

{

return 1;

}

**YACC Code**

/\*

Y File

Implementing parser using lex and yacc to validate <img> and <a> tag of html

\*/

%{

#include<stdio.h>

#include<stdlib.h>

int yylex();

int yyerror();

%}

//tokens generated by lexical analyzer and passed to parser.

%token START\_ANGLE\_BRACKET NL A\_TAG A\_ATTRIBUTE SYNTAX\_ERROR EQUAL ATTR\_VALUE END\_TAG1 A\_CONTENT A\_END\_TAG IMG\_TAG IMG\_ALT IMG\_SRC IMG\_OTHER\_ATTRIBUTE IMG\_END\_TAG2 IMG\_END\_TAG3

%%

//Grammer rules.

//if stmt is reached msg valid syntax is printed.

stmt : S NL { printf("\n Valid syntax ...\n\n");

exit(0);}

;

//start symbol(S) goes to <img> tag or <a> tag.

// START\_ANGLE\_BRACKET = "<".

S : START\_ANGLE\_BRACKET A\_TAG anchor\_attrs

| START\_ANGLE\_BRACKET IMG\_TAG img\_comp\_attr img\_other\_attr img\_ending

;

img\_comp\_attr : img\_comp\_attr\_1

| img\_comp\_attr\_2

;

img\_comp\_attr\_1 : IMG\_SRC IMG\_ALT

;

img\_comp\_attr\_2 : IMG\_ALT IMG\_SRC

;

img\_other\_attr : IMG\_OTHER\_ATTRIBUTE img\_other\_attr

|

;

//img can have three type of end tag >, /> or </img>.

img\_ending : END\_TAG1

| IMG\_END\_TAG2

| IMG\_END\_TAG3

;

anchor\_attrs : A\_ATTRIBUTE anchor\_attrs

| END\_TAG1 anchor\_rem A\_END\_TAG

;

//Tags nesting is done here in <a> tag.

anchor\_rem : content S anchor\_rem

| content

;

content : A\_CONTENT content

|

;

%%

void main()

{

printf("\n Enter the string : ");

yyparse();

}

int yyerror(char \*msg)

{

printf("\n Invalid syntax : Error : %s\n\n",msg);

exit(0);

}

##### References

Compilers, principles, techniques, and tools / Alfred V. Aho, Ravi. Sethi, Jeﬀrey D. Ullman. 1986. ISBN 0-321-48681-1

<https://www.geeksforgeeks.org/> for tag syntax

THANK YOU