Compiler Assignment:

Submission

1. Language Features

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Example** |
| Keywords | The various keywords present in the language include ’if’, ’return’, ’for’, ’building’, ’floor’, ’.start’, ’.end’ |  |
| Identifiers | Identifiers in the language are of the form [\_a-zA-Z][\_a-zA-Z0-9]\* ie. They begin with an alphabetical character followed by an alphanumeric character. |  |
| Data types | The data types supported in our language are int, float, distance, point, wall, door, window and ratio. |  |
| Operators | **‘=’** is the Assignment operator.  ‘–‘operator performs subtraction operation.  ‘\*’ operator computes simple multiplication of two numbers specified around it.  ‘+’ operator computes simple addition of two numbers. Numbers may be integer or float.  ‘||’thickness operator.  ‘&’operator |  |
| Function | A procedure can be called with any number of parameters of available data types , and only one return data type. In the example shown Wall is return type of procedure. | Wall function(Wall w1,Point p2) {  } |
| Scope Rules |  |  |
| Conditional Statements | Our language supports if construct, with conditions being checked via composition of relational operators (<,<=,>,>=,==) and identifiers. | if(a==b) {  } |
| Iterative Statements | The language supports a ‘for’ loop design as described in the example on the right. In the example shown count is an integer type variable. | for(count=1;count<12;count=count+1) |
| **Name** | **Description** | **Example** |
| I/O Operations | Since our language is a domain specific language that writes programs only for the purpose of drawing, therefore the programs written in our language do not take inputs, hence no Input operations. Also the output has to be given by default , as such there is no output operator as well. |  |
| Expression | Expressions are of the forms as shown in the examples to the right. | i=i+1 or i=i-1 or i=i\*I or i=i/I or W1=t||(p1&p2); |
| Assignment Statements | Different data types have different types of assignment statements. However the assignment operator is always ‘=’.Examples of various assignment statements have been shown on the right. | ratio r = 3:4;  point p1=(3,4);  point P1=w1.start;  point P1=w1./end;  Point p1=(w1,w1.start,d);  Wall W1=t||(p1&p2);  Wall W1=t||(w2,p1,theta1)  Wall w1=(w2,w2.end,90,14);  Door d1= (p1,p2); |

1. Lexical Units: The following are the lexical units in our programming language:

|  |  |  |
| --- | --- | --- |
| **Pattern** | **Token** | **Purpose** |
| EPS | TK\_EPSILON | Epsilon |
| ( | TK\_ROUND\_OPEN | Delimiter |
| ) | TK\_ROUND\_CLOSE | Delimiter |
| { | TK\_CURLY\_OPEN | Delimiter |
| } | TK\_CURLY\_CLOSE | Delimiter |
| , | TK\_COMMA | Delimiter |
| int | TK\_INT | Datatype |
| float | TK\_FLOAT | Datatype |
| distance | TK\_DISTANCE | Datatype |
| point | TK\_POINT | Datatype |
| wall | TK\_WALL | Datatype |
| door | TK\_DOOR | Datatype |
| window | TK\_WINDOW | Datatype |
| ratio | TK\_RATIO | Datatype |
| for | TK\_FOR | Keyword “for” |
| ; | TK\_SEMICOLON | Delimiter |
| if | TK\_IF | Keyword “if” |
| return | TK\_RETURN | Keyword “return” |
| = | TK\_EQUALTO | Assignment Operator |
| [ | TK\_SQUARE\_OPEN | Delimiter |
| ] | TK\_SQUARE\_CLOSE | Delimiter |
| : | TK\_COLON | Ratio Operator |
| || | TK\_THICKNESS | Thickness Operator |
| .start | TK\_DOTSTART | Keyword “.start” |
| .end | TK\_DOTEND | Keyword “.end” |
| & | TK\_AND | Point Operator |
| \* | TK\_MUL | Multiply Operator |
| - | TK\_MINUS | Subtraction Operator |
| + | TK\_PLUS | Addition Operator |
| / | TK\_DIVIDE | Division Operator |
| % | TK\_MODULO | Modulo Operator |
| BUILDING | TK\_BUILDING | Keyword “building” |
| floor | TK\_FLOOR | Keyword “floor” |
| < | TK\_LESS\_THAN | Conditional Operator |
| > | TK\_GREATER\_THAN | Conditional Operator |
| <= | TK\_LESS\_THAN\_EQUAL\_TO | Conditional Operator |
| >= | TK\_GREATER\_THAN\_EQUAL\_TO | Conditional Operator |
| == | TK\_EQUAL\_TO | Conditional Operator |
| != | TK\_NOT\_EQUAL\_TO | Conditional Opeator |
| [\_a-zA-Z][\_a-zA-Z0-9]\* | TK\_IDENTIFIER | Alphabetical Characters |
| [-+]?[0-9]\*\.?[0-9]+ | TK\_LITERAL | Rational Number |

1. LL(1) Grammar:

|  |
| --- |
|  |

1. Test Cases:
   1. Test Case 1:

|  |
| --- |
| building b{  floor f1{  point p1 = (0,0);  point p2 = (100,0);  wall w1 = 2||(p1 & p2);  wall w2 = 3||(w1,p2,90,40);  point p4 = w2.end;  point p3 = (p4,w2,5);  window win1 = (p1,p2);  door d1 = (p3,p4);  }  } |

* 1. Test Case 2:

|  |
| --- |
| wall func(wall w, ratio r, float d,){  point p3 = w.start;  point p=(w,p3,r);  wall w1 = 2||(w,p,90,d);  return w1  }  building b{  floor f {  point p1 = (0,0);  point p2 = (25,0);  point p3 = (25,25);  point p4 = (0,25)  wall bound = 2||(p1 & p2);  wall w;  ratio r = 1:2;  w = func(bound,r,10,);    }  } |

* 1. Test Case 3:

|  |
| --- |
| building b1{  floor f1{  point p1 = (0,0);  Point p2 = (100,0);  wall w1 = 2||(p1 & p2);  wall w2 = 3||(w1,p2,90,40);    }  floor f2  {  point p3 = (30,0);  point p4 = (70,0);  wall w3 = 2||(p3 & p4);  }  } |

* 1. Test Case 4:

|  |
| --- |
| building b{  floor f{      int i = 1;  point p1 = (0,0);  point p2 = (25,0);  point p3 = (5,0);  wall bound= 2||(p1 & p2);  if(i==1)  {  door d = (p1,p3);  }  }  } |

* 1. Test Case 5:

|  |
| --- |
| Building b{  Floor f{  wall W[4];  int i =0;    for(i = 0 ;i<4;i = i+1){  int var\_a = 2\*i;    point pt\_d = (0,var\_a);  point pt\_e = (25,var\_a);  w[i] = 2||(pt\_d & pt\_e);    }  }  } |