Compiler Construction
Assignment - 2
Design Specs and SDT Design

Group Details

Group Name - Tetreasy

Group Number - 39

Group Members:

Aman Sharma	2019A7PS0053G
Aarjav	2019A7PS0204G
Rudra Pratap Singh Chouhan	2019A7PS0164G
Shashank Gupta	2019A7PS0029G
Daksh Talreja	2019A7PS0014G
Ketan Raghunath Raut	2019A7PS0150G

STAGE 1

Chapter 1
Top Level Design Specifications

1. **Statement**

   The programming language we have designed tries to provide the programmer with the flexibility in terms of the problem domain.
   The aim is to enable the programmer to efficiently design the game with multiple features providing a sense of independence and control to the programmer, thereby not losing out on too much jargon at the same time.

2. **Assumptions**

   We have been tasked with the responsibility of creating a programming language. For the ease of conversation and relatability from here on, we will name them and refer to them accordingly.

   Name of the programming language - Tetreasy
   Name of the programmer(user) - Sarah

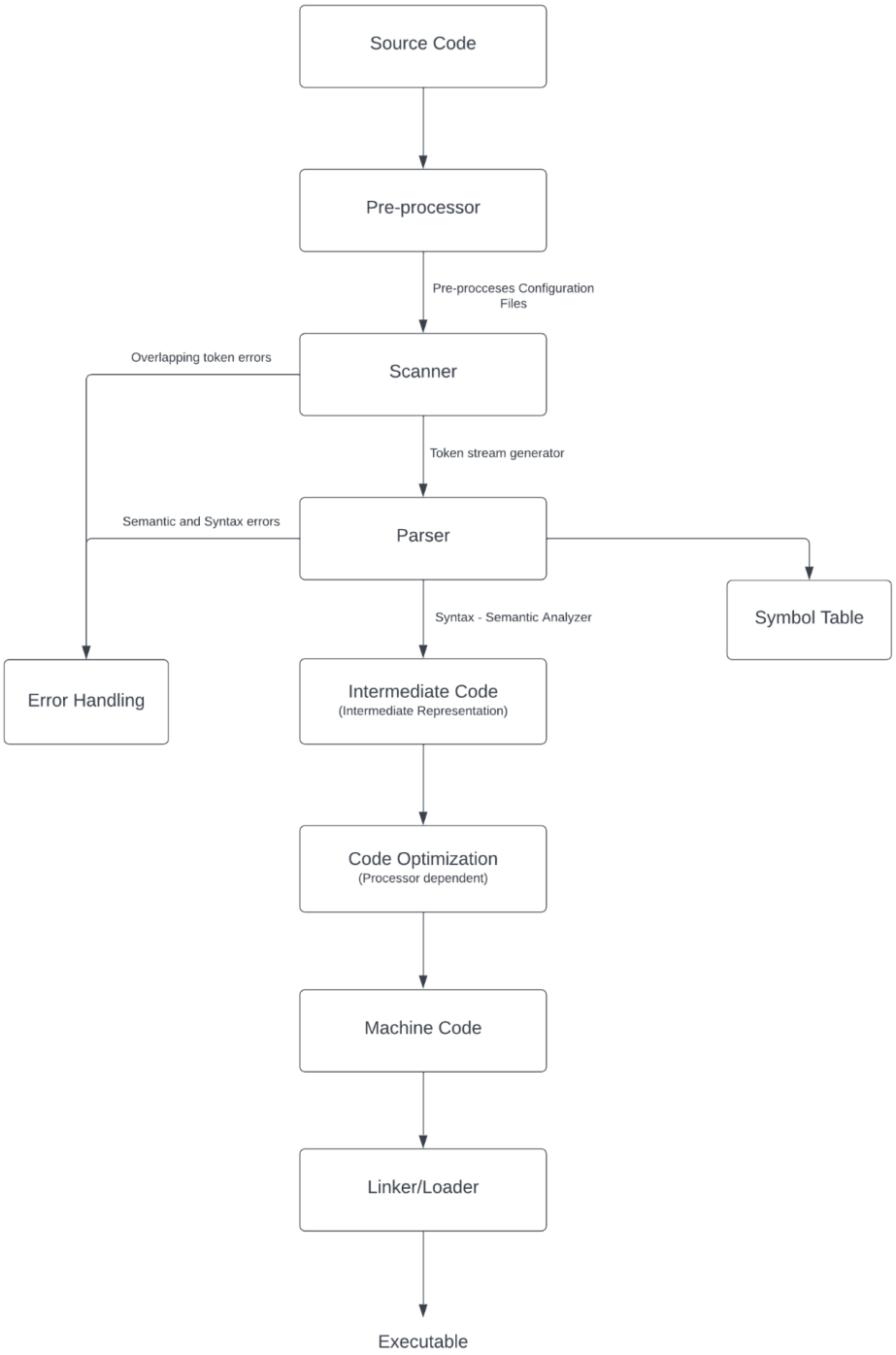
3. Overall Structure

Code written in Tetreasy will be scanned and parsed to create the target python file which can then be executed to play the game. Tetreasy uses space to distinguish between tokens, which would be acting as the delimiter.

4. Offered primitives/Features

- Rows: This would enable Sarah to set the number of rows in the tetris grid.
- Columns: This would enable Sarah to set the number of columns in the tetris grid.
- Config: This helps in setting the keys corresponding to movement options.
- Block :Block defines if a blocktype will be present in the game or not. Moreover, it could be used to set its color as well.
- Speed: This can be used to set the speed of dropping pieces.
  This would enable Sarah to alter the difficulty level of the game.
- timedgame: The value of timedgame can be set to True or False depending on the requirements of timer in game.
- timer: This feature sets up the amount of time to be survived to win the game. For example, if the value of timer is set to 20; it would mean that the player needs to survive a total of 20 seconds in order to win the game.
- music: Suggestively, this aids in selecting the music to be played.
- nextq: This is used to set the number of blocks visible in the next-queue.

5. Pipeline Schema



Steps in the Pipeline Schema

- First, the code in Tetreasy will be sent to a scanner. Scanner will then return tokens of the input stream.
- Parser makes the Abstract Syntax Tree from the tokens provided.
- Semantic Analyzer checks for syntactical errors.
- After checking the program for syntax errors, an intermediate program would be generated from the code provided by the programmer in Tetreasy.
- An executable is then made from this code which will finally be able to run the game of Tetris.

Lexical Analyser/Scanner -The scanner is a subroutine which is frequently called by an application program like a compiler. The primary function of a scanner is to combine characters from the input stream into recognizable units called tokens. Overall, scanning is the process of reading the source code one character at a time in a methodical manner to convert them into tokens**.**

Syntax analyzer / Parser:
Parsing is the process of taking the tokens and generating a parse tree as the output. Parser is a program that does the parsing.

Semantic analyzer: semantic analysis is the process of drawing meaning from text. It allows computers to understand and interpret sentences, paragraphs, or whole documents, by analyzing their grammatical structure, and identifying relationships between individual words in a particular context

Target code generation: Target code generation is the last stage of compilation. Each line in optimized code may map to one or more lines in machine/assembly code.

Chapter 2
Scanner Design

The table showcasing various lexemes, the token names and their attribute values.

Lexeme	Token Name	Attribute Value
=	assinop	AS
ID	IDENTIFIER	[a-zA-Z_][a-zA-Z0-9_]*
Any number	number	-
CONFIG	CONFIG	CONFIG
ROWS	KEYWORD	ROWS
COLUMN	KEYWORD	COLUMN
MUSIC	KEYWORD	MUSIC
SPEED	KEYWORD	SPEED
NEXTQ	KEYWORD	NEXTQ
ORIGIN	KEYWORD	ORIGIN
BLOCK1	BLOCK	BLOCK1
BLOCK2	BLOCK	BLOCK2
BLOCK3	BLOCK	BLOCK3
BLOCK4	BLOCK	BLOCK4
BLOCK5	BLOCK	BLOCK5
BLOCK6	BLOCK	BLOCK6

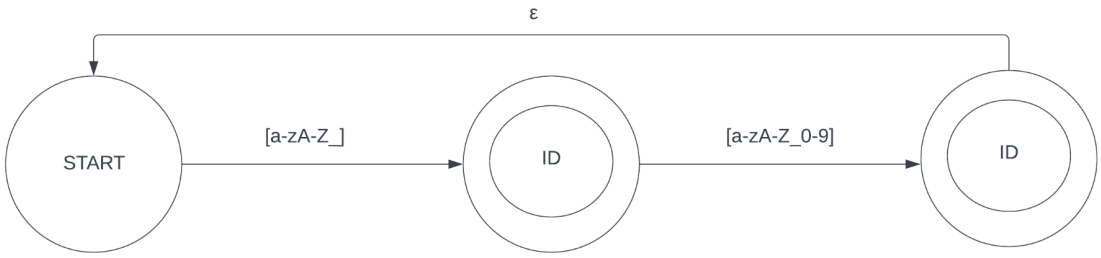
BLOCK7	BLOCK	BLOCK7
TIMER	KEYWORD	TIMER
TIMEDGAME	KEYWORD	TIMEDGAME
TRUE	BOOLEAN	TRUE
FALSE	BOOLEAN	FALSE
LEFT	MOVE	LEFT
RIGHT	MOVE	RIGHT
SOFTDROP	MOVE	SOFTDROP
HARDDROP	MOVE	HARDDROP
ROTATELEFT	MOVE	ROTATELEFT
ROTATERIGHT	MOVE	ROTATERIGHT
PAUSE	COLOR	PAUSE
RED	COLOR	RED
BLUE	COLOR	BLUE
GREEN	COLOR	GREEN
YELLOW	COLOR	YELLOW
ORANGE	COLOR	ORANGE
INDIGO	COLOR	INDIGO
BLACK	COLOR	BLACK
RANDOM	COLOR	RANDOM

Transition Diagrams

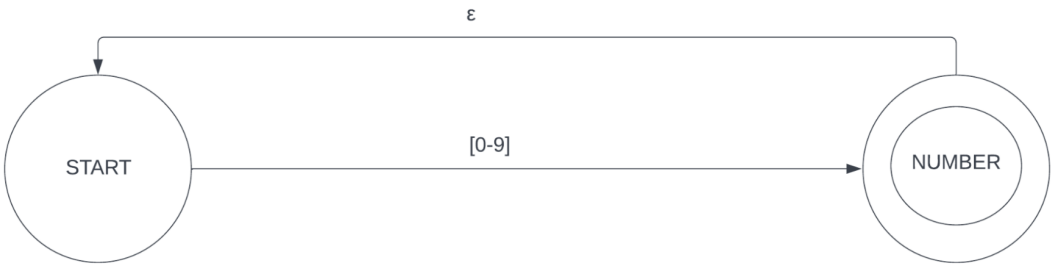
-> START STATE: The Start state is denoted by a starting state where the system enters for reading the lexemes from the source code provided.

-> FINAL STATE: The final state is denoted with a double circle with the token name.

Transition Diagram of Identifier:



Transition Diagram of Numbers:



Chapter 3
Division of Labor between the Scanner and Parser

We are using the scanner to identify errors in the tokens. The type of token is identified by the scanner. The parser creates the semantic trees.

Our scanner will take a sequence of characters (the source file of the program) and produce a sequence of tokens that will be used to feed the compiler's parser. It will consist of NEXTQ, LEFT, RIGHT, ROTATELEFT, HARDDROP, SOFTDROP, etc.

Taking examples:

LEFT	-	Key assigned to it will shift the incoming block left.
RIGHT	-	Key assigned to it will shift the incoming block right.
ROTATELEFT	-	Key assigned to it will rotate the blocks.
HARDDROP	-	Key assigned to it will force push down the incoming blocks vertically.
SOFTDROP	-	Key assigned to it will push down the incoming blocks by 1 row.

Parser will interpret the component that breaks data into smaller elements for easy translation into another language. Our parser will do the syntax analysis, this process is called parsing. The parser checks whether the expression made by the tokens is syntactically correct or not.

Note: We have not defined the “keys” as a token type. Instead, we use identifiers to read keys. The task of identifying whether a key is legal or not is done by the parser. The parser has a dictionary with names of keyboard keys mapped to their corresponding values in pygame. This way, instead of checking the names of keys with the scanner and then assigning value during parsing, we perform both of these tasks during parsing.

Chapter 4

Division and Distribution of Roles and Responsibilities Among the team

The team has made combined efforts to bring the project to fruition.
The team has worked in such a way that everyone contributed significantly to produce the final outcome in the form of Tetreasy for Sarah to work on.
The work was distributed among the team members as follows:

1. Aman Sharma
    - Parser Design & code
    - tetris.py code
2. Aarjav
    - Grammar generation
    - Reporting and maintenance
3. Rudra Pratap Singh Chouhan
    - Syntax Directed Translation scheme
4. Shashank Gupta
    - Test Cases and their purposes
5. Daksh Dilip Talreja
    - Creation of the Python Virtual Environment
    - Scanner Design & code
6. Ketan Raghunath Raut
    - nextQ Feature Implementation

STAGE 2

Chapter 5
Syntax Directed Translation Scheme

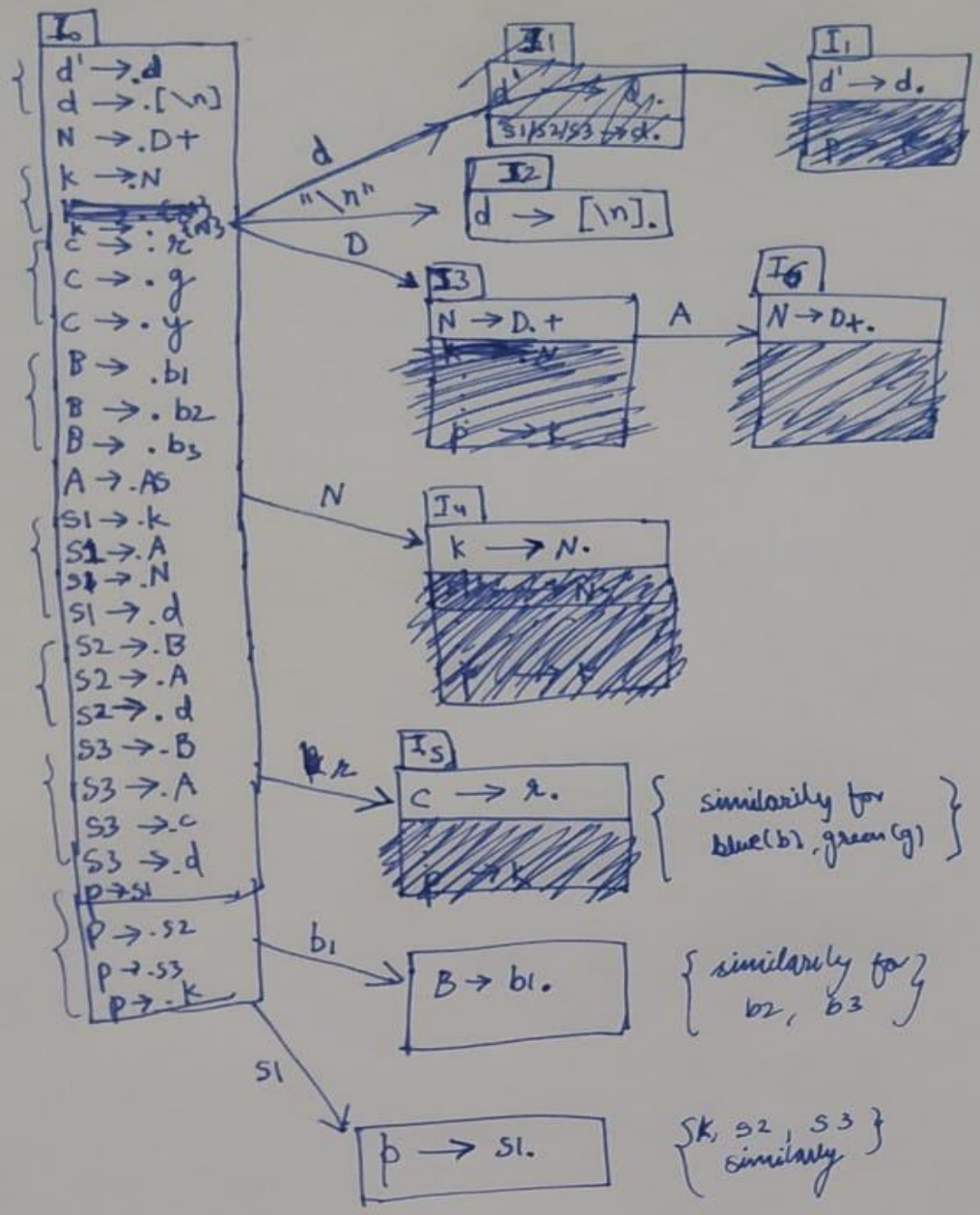
In syntax directed translation, along with the grammar we associate some informal notations and these notations are called semantic rules.

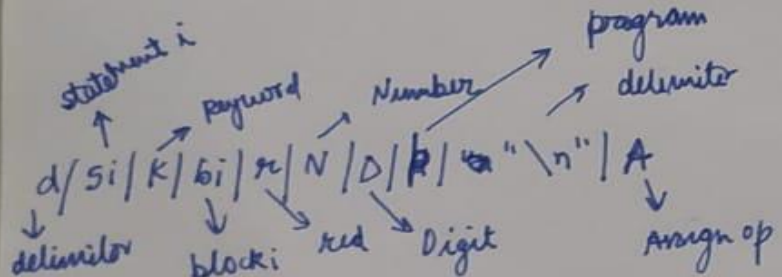
So we can say that

- Grammar + semantic rule = SDT (syntax directed translation)
- The Syntax directed translation scheme is a context -free grammar.
- The syntax directed translation scheme is used to evaluate the order of semantic rules.
- In the translation scheme, the semantic rules are embedded within the right side of the productions.
- The position at which an action is to be executed is shown by enclosed between braces. It is written within the right side of the production.

LR(0) AUTOMATON

LR(0)


I0
d' → .d
d → .[\n]
N → .D+
k → .N
c → .r
c → .g
c → .y
B → .b1
B → .b2
B → .b3
A → .AD
S1 → .k
S1 → .A
S1 → .N
S1 → .d
S2 → .B
S2 → .A
S2 → .d
S3 → .B
S3 → .A
S3 → .c
S3 → .d
p → .S1
p → .S2
p → .S3
p → .k
d
I1
d' → d.
p → d.
"\n"
I2
d → [\n].
D
I3
N → D.+
A
I6
N → D+.
N
I4
k → N.
r
I5
C → r.
similarly for blue(b), green(g)
b1
B → b1.
similarly for b2, b3
S1
p → S1.
k, S2, S3 similarly


statement i
Keyword
Number
program delimiter
d / Si / K / bi / r / N / D / p / "\n" / A
delimiter
blocki
red
Digit
Assign op

Chapter 6
An Explanation of Challenges

- Throughout the process of making the project, many challenges were encountered. A shift-reduce conflict was noticed and tackled effectively when we removed the dangling else pointers from the grammar. The Shift-Reduce Conflict is the most common type of conflict found in grammars. It occurs in a state that requests both a shift action and a reduced action.

- In the case of errors and error handling, we have encountered quite a few errors such as collision etc, and they were resolved by manual embedding actions specifically aimed for error handling. It is mentioned how exactly the example errors are tackled to continue the game without any problem even after encountering it.

1. **In the case of the columns, if the number of columns gets below 4, it shows an error. (similarly for rows)**

```
if "COLUMNS" in parser.names:
    COLUMNS=parser.names["COLUMNS"]
    if(COLUMNS<4):
        print("Error: COLUMNS cannot be less than 4.")
        return
    print("COLUMNS updated")
else:
    print("default value assigned for COLUMNS")
```

2. **If ORIGIN+3 >= columns or ORIGIN < 0, it gives an error.**

```
if "ORIGIN" in parser.names:
    ORIGIN=parser.names["ORIGIN"]
    if(ORIGIN<0 or ORIGIN+3>=COLUMNS):
        print("Error: ORIGIN cannot be less than 0 or greater than COLUMNS-4.")
        return
    print("ORIGIN updated")
else:
    print("default value assigned for ORIGIN")
```

3. **If the value assigned to MUSIC is greater than 2 the program will give an error message. If no value is assigned to MUSIC , the game will run with the default value of music. Similarly if the value of speed assigned by the programmer is out of bounds ,an error message will be displayed.**

```
    if "MUSIC" in parser.names:
            MUSIC=parser.names["MUSIC"]
            if(MUSIC<0 or MUSIC>2):
                print("Error: MUSIC cannot be less than 0 or greater
than")
                return
            print("MUSIC updated")
        else:
            print("default value assigned for MUSIC")
```

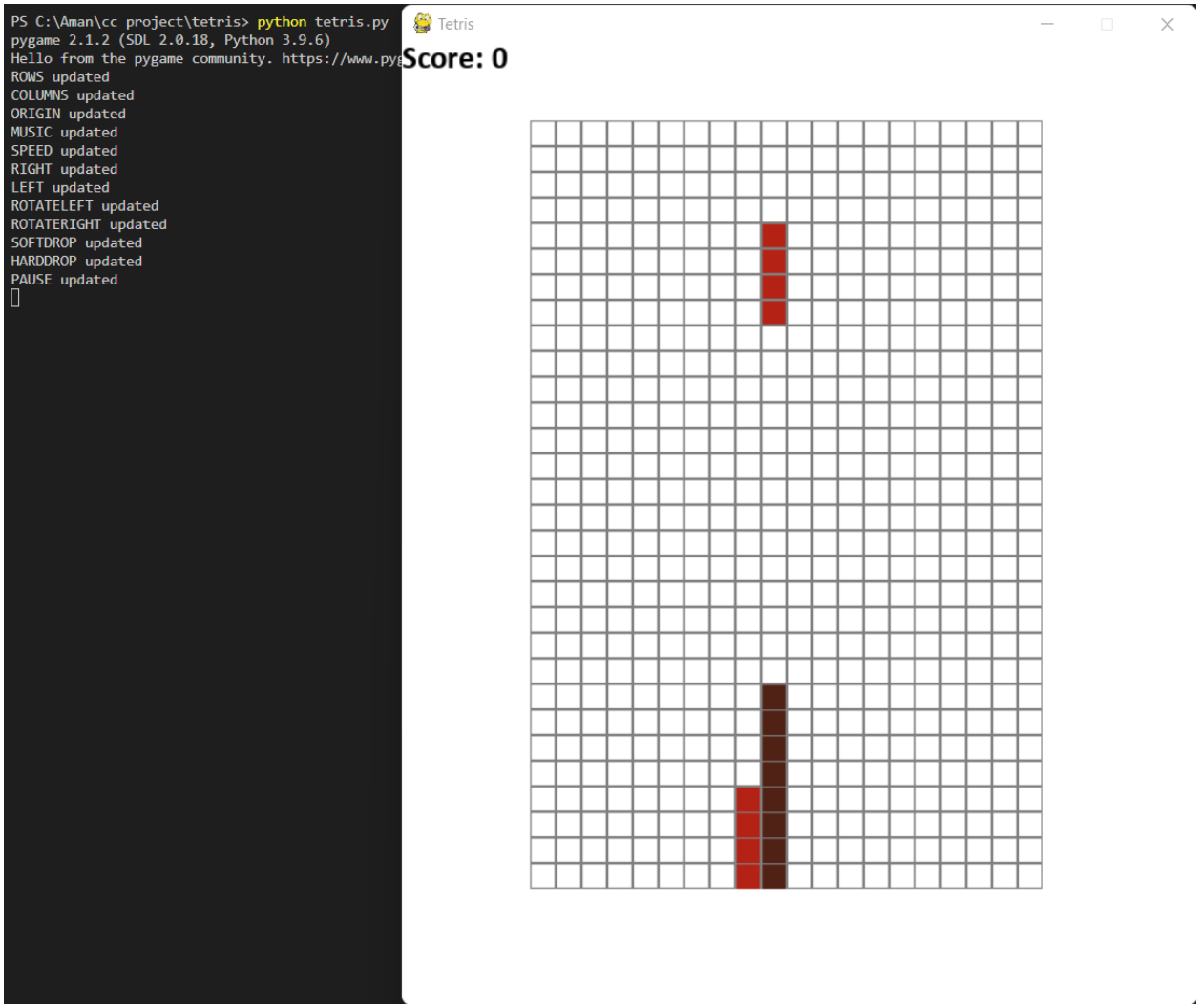
Chapter 7
Test Cases and their purposes

Test Case 1:

Basic example where a programmer writes .tet code to set up all necessary parameters such as rows, columns, etc. ; assigns key bindings, and sets colors and T/F values for all blocktypes.

program.tet 452 bytes

```
ROWS = 30
COLUMNS = 20
ORIGIN = 8
MUSIC = 1
SPEED = 2
NEXTQ = 1

BLOCK1 = TRUE
BLOCK2 = FALSE
BLOCK3 = FALSE
BLOCK4 = FALSE
BLOCK5 = FALSE
BLOCK6 = FALSE
BLOCK7 = FALSE

BLOCK1 = RANDOM
BLOCK2 = RED
BLOCK3 = BLACK
BLOCK4 = BLUE
BLOCK5 = GREEN
BLOCK6 = YELLOW
BLOCK7 = RANDOM

// aknflesknfpianefpioe

CONFIG RIGHT K_RIGHT
CONFIG LEFT K_LEFT
CONFIG ROTATELEFT K_SPACE
CONFIG ROTATERIGHT K_RETURN
CONFIG SOFTDROP K_s
CONFIG HARDDROP K_h
CONFIG PAUSE K_p
```


PS C:\Aman\cc project\tetris> python tetris.py
pygame 2.1.2 (SDL 2.0.18, Python 3.9.6)
Hello from the pygame community. https://www.pyg
ROWS updated
COLUMNS updated
ORIGIN updated
MUSIC updated
SPEED updated
RIGHT updated
LEFT updated
ROTATELEFT updated
ROTATERIGHT updated
SOFTDROP updated
HARDDROP updated
PAUSE updated
Tetris
Score: 0

Test Case 2:

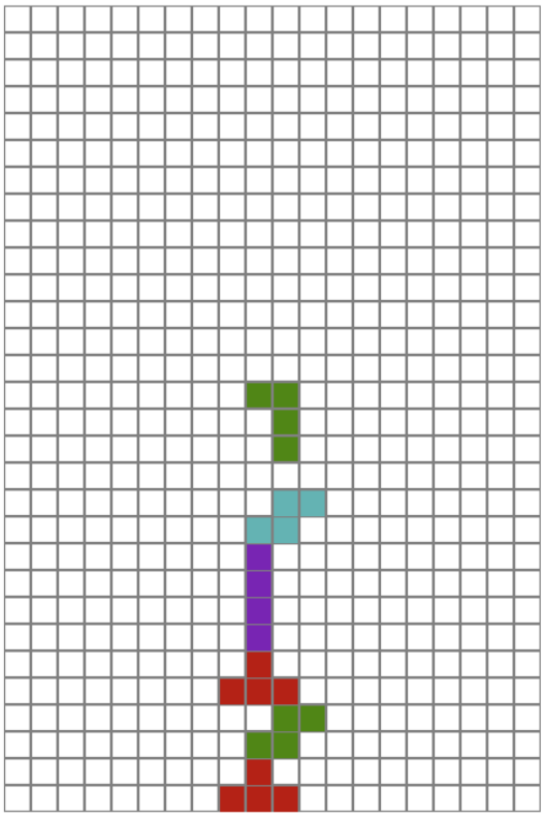
When a programmer skips assignment of some attributes, tetris.py provides default values to be initialized.

```
ROWS = 30
COLUMNS = 20
ORIGIN = 8
MUSIC = 1
NEXTQ = 1

BLOCK1 = TRUE
BLOCK2 = FALSE
BLOCK7 = FALSE

BLOCK1 = RANDOM
BLOCK2 = RED


// Key Bindings

CONFIG RIGHT K_RIGHT
CONFIG LEFT K_LEFT
CONFIG ROTATELEFT K_SPACE
CONFIG ROTATERIGHT K_RETURN
CONFIG PAUSE K_p
```

```
PS C:\Aman\cc project\tetris> python tetris.py
pygame 2.1.2 (SDL 2.0.18, Python 3.9.6)
Hello from the pygame community. https://www.py
ROWS updated
COLUMNS updated
ORIGIN updated
MUSIC updated
SPEED updated
RIGHT updated
LEFT updated
ROTATELEFT updated
ROTATERIGHT updated
SOFTDROP updated
HARDDROP updated
PAUSE updated
```

Tetris

Score: 0



Test Case 3:

If the programmer sets all blocks as FALSE, in all the block types ,the program won't run and will ask the programmer to set at least one of the blocks as TRUE.

```
BLOCK1 = FALSE
BLOCK2 = FALSE
BLOCK3 = FALSE
BLOCK4 = FALSE
BLOCK5 = FALSE
BLOCK6 = FALSE
BLOCK7 = FALSE
```

OUTPUT: `"Error: Atleast one block type has to be allowed"`

```
PS C:\Aman\cc project\tetris> python tetris.py
pygame 2.1.2 (SDL 2.0.18, Python 3.9.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
ROWS updated
COLUMNS updated
ORIGIN updated
MUSIC updated
SPEED updated
Error: Atleast one block type has to be allowed
```

Chapter 8
A complete end-to-end Tetris game engine programming toolchain

The program is compiled and run with a single command, i.e. “python tetris.py”. On running this command, the tetris.py program first opens the program file, i.e. the file containing the code written in our tetris programming language. It is read line by line and stored in an array of strings.

Next, it creates an instance of the class “ TetrisLexer " contained in the file “tetrisscanner.py”. The instance of the scanner is fed the game code line by line. The scanner tokenizes the code and show error message for any unacceptable tokens.

tetrisscanner.py:

```
from sly import Lexer

class TetrisLexer(Lexer):
    # Set of token names.
    tokens = { KEYWORD, NUMBER, MOVE, ASSIGNOP,BOOLEAN,DELIMITER, ID, CONFIG, BLOCK, COLOR}

    # String containing ignored characters between tokens
    ignore = ' \t'

    # Regular expression rules for tokens
    ID = r'[a-zA-Z_][a-zA-Z0-9_]*'
    ID['TRUE'] = 'BOOLEAN'
    ID['FALSE'] = 'BOOLEAN'
    ID['CONFIG'] = 'CONFIG'
    ID['ROWS'] = 'KEYWORD'
    ID['COLUMNS'] = 'KEYWORD'
    ID['ORIGIN'] = 'KEYWORD'
    ID['MUSIC'] = 'KEYWORD'
    ID['SPEED'] = 'KEYWORD'
    ID['NEXTQ'] = 'KEYWORD'
    ID['BLOCK1'] = 'BLOCK'
    ID['BLOCK2'] = 'BLOCK'
    ID['BLOCK3'] = 'BLOCK'
    ID['BLOCK4'] = 'BLOCK'
    ID['BLOCK5'] = 'BLOCK'
    ID['BLOCK6'] = 'BLOCK'
    ID['BLOCK7'] = 'BLOCK'
    ID['TIMER'] = 'KEYWORD'
    ID['TIMEDGAME'] = 'KEYWORD'
    ID['RIGHT'] = 'MOVE'
    ID['LEFT'] = 'MOVE'
    ID['SOFTDROP'] = 'MOVE'
    ID['HARDDROP'] = 'MOVE'
    ID['ROTATERIGHT'] = 'MOVE'
    ID['ROTATELEFT'] = 'MOVE'
    ID['PAUSE'] = 'MOVE'
    ID['RED'] = 'COLOR'
    ID['BLUE'] = 'COLOR'
    ID['GREEN'] = 'COLOR'
    ID['YELLOW'] = 'COLOR'
    ID['ORANGE'] = 'COLOR'
    ID['INDIGO'] = 'COLOR'
    ID['VIOLET'] = 'COLOR'
    ID['BLACK'] = 'COLOR'
    ID['RANDOM'] = 'COLOR'
    NUMBER  = r'\d+'
    ASSIGNOP  = r'='
    DELIMITER = r'\n'

    @_(r'//.*')
    def COMMENT(self, t):
        pass

    def error(self, t):
        print('Line %d: Bad character %r' % (self.lineno, t.value[0]))
        self.index += 1
```

The tokens returned by the scanner are then passed to the parser, again on a line by line basis. The parser analyzes the syntax and rejects any syntactically incorrect statements. The information from parsed statements is stored in appropriate data structures (dictionaries and sets).

tetrisparser.py:

```
from sly import Parser
from tetrisscanner import TetrisLexer

dictofkeys = {'K_BACKSPACE': 8, 'K_TAB': 9, 'K_CLEAR': 1073741980, 'K_RETURN': 13, 'K_PAUSE': 1073741896, 'K_ESCAPE': 27,
'K_SPACE': 32, 'K_EXCLAIM': 33, 'K_HASH': 35, 'K_QUOTEDBL': 34, 'K_DOLLAR': 36, 'K_AMPERSAND': 38, 'K_QUOTE': 39, 'K_LEFTPAREN': 40,
'K_RIGHTPAREN': 41, 'K_ASTERISK': 42, 'K_PLUS': 43, 'K_COMMA': 44, 'K_MINUS': 45, 'K_PERIOD': 46, 'K_SLASH': 47, 'K_0': 48, 'K_1': 49,
'K_2': 50, 'K_3': 51, 'K_4': 52, 'K_5': 53, 'K_6': 54, 'K_7': 55, 'K_8': 56, 'K_9': 57, 'K_COLON': 58, 'K_SEMICOLON': 59, 'K_LESS': 60,
'K_EQUALS': 61, 'K_GREATER': 62, 'K_QUESTION': 63, 'K_AT': 64, 'K_LEFTBRACKET': 91, 'K_RIGHTBRACKET': 93, 'K_CARET': 94, 'K_UNDERSCORE': 95,
'K_a': 97, 'K_b': 98, 'K_c': 99, 'K_d': 100, 'K_e': 101, 'K_f': 102, 'K_g': 103, 'K_h': 104, 'K_i': 105, 'K_j': 106, 'K_k': 107, 'K_l': 108,
'K_m': 109, 'K_n': 110, 'K_o': 111, 'K_p': 112, 'K_q': 113, 'K_r': 114, 'K_s': 115, 'K_t': 116, 'K_u': 117, 'K_v': 118, 'K_w': 119, 'K_x': 120,
'K_y': 121, 'K_z': 122, 'K_DELETE': 127, 'K_UP': 1073741906, 'K_DOWN': 1073741905, 'K_RIGHT': 1073741903, 'K_LEFT': 1073741904, 'K_INSERT': 1073741897,
'K_HOME': 1073741898, 'K_END': 1073741901, 'K_PAGEUP': 1073741899, 'K_PAGEDOWN': 1073741902, 'K_NUMLOCK': 1073741907, 'K_CAPSLOCK': 1073741881,
'K_SCROLLOCK': 1073741895, 'K_RSHIFT': 1073742053, 'K_LSHIFT': 1073742049, 'K_RCTRL': 1073742052, 'K_LCTRL': 1073742048, 'K_RALT': 1073742054, 'K_LALT': 1073742050}

class TetrisParser(Parser):

    tokens = TetrisLexer.tokens

    def __init__(self):
        self.names = {}
        self.blocktable = [True, True, True, True, True, True, True]
        self.colortable = ["RANDOM","RANDOM","RANDOM","RANDOM","RANDOM","RANDOM","RANDOM"]

    @_('KEYWORD ASSIGNOP NUMBER DELIMITER')
    def statement(self,p):
        self.names[p.KEYWORD]=int(p.NUMBER)

    @_('BLOCK ASSIGNOP BOOLEAN DELIMITER')
    def statement(self,p):
        # self.names[p.BLOCK]=p.BOOLEAN
        if(p.BLOCK=="BLOCK1"):
            if(p.BOOLEAN=="FALSE"):
                self.blocktable[0]=False
        if(p.BLOCK=="BLOCK2"):
            if(p.BOOLEAN=="FALSE"):
                self.blocktable[1]=False
        if(p.BLOCK=="BLOCK3"):
            if(p.BOOLEAN=="FALSE"):
                self.blocktable[2]=False
        if(p.BLOCK=="BLOCK4"):
            if(p.BOOLEAN=="FALSE"):
                self.blocktable[3]=False
        if(p.BLOCK=="BLOCK5"):
            if(p.BOOLEAN=="FALSE"):
                self.blocktable[4]=False
        if(p.BLOCK=="BLOCK6"):
            if(p.BOOLEAN=="FALSE"):
                self.blocktable[5]=False
        if(p.BLOCK=="BLOCK7"):
            if(p.BOOLEAN=="FALSE"):
                self.blocktable[6]=False
```

```
@_('BLOCK ASSIGNOP COLOR DELIMITER')
def statement(self,p):
    if(p.BLOCK=="BLOCK1"):
        self.colortable[0]=p.COLOR
    elif(p.BLOCK=="BLOCK2"):
        self.colortable[1]=p.COLOR
    elif(p.BLOCK=="BLOCK3"):
        self.colortable[2]=p.COLOR
    elif(p.BLOCK=="BLOCK4"):
        self.colortable[3]=p.COLOR
    elif(p.BLOCK=="BLOCK5"):
        self.colortable[4]=p.COLOR
    elif(p.BLOCK=="BLOCK6"):
        self.colortable[5]=p.COLOR
    elif(p.BLOCK=="BLOCK7"):
        self.colortable[6]=p.COLOR


@_('CONFIG MOVE ID DELIMITER')
def statement(self,p):
    if p.ID not in dictofkeys:
        print(p.ID + " is an invalid key. Using default value")
        pass
    self.names[p.MOVE]=dictofkeys[p.ID]

@_('DELIMITER')
def statement(self,p):
    pass

def error(self, p):
    print("Error encountered during parsing")
```

The information retrieved by the program is then used by the tetris program to determine the configurations.