PYTHON PROJECT

GAMES

Done by:
Rashi Jhawar

SYSTEM SPECIFICATION

a) Hardware Requirement

Processor: Pentium processor with Windows OS

Memory: 512 MB

Disk Capacity: 1 GB

b) Software Requirement

Python 2.7

PROJECT OVERVIEW

The project includes the three well-renowned games made to work using a blend of Python user-defined functions and classes, namely:

a) Word search

In the Word search game, a number of letters of words are placed in a grid, which usually has a square shape. The objective of this puzzle is to find and mark all the words hidden inside the box. This is done by entering the coordinates of the starting and the ending letter during program run. The words may be placed horizontally, vertically, or diagonally.

b) Hangman

In the Hangman game, the player has to guess the word chosen randomly by the computer within a specific number of trials. The word to guess is represented by a row of dashes, giving the number of letters, numbers and category. If the guessing player suggests a letter or number which occurs in the word, the computer writes it in all its correct positions. If the suggested letter or number does not occur in the word, the computer draws one element of a hanged man stick figure as a tally mark. The game gets over when the player guesses the whole word correctly (player wins the game) or the computer completes the diagram (player loses the game).

c) Tic-Tac-Toe

In the Tic-Tac-Toe game, the player plays against a simple artificial intelligence. An artificial intelligence (or AI) is a computer program that can intelligently respond to the player's moves. Tic-Tac-Toe is a simple game to play with a paper and pencil between two people. One player is X and the other player is O. On a simple nine square grid (which we call the board), the players take turns placing their X or O) on the board. If a player gets three of their marks on the board in a row, column or one of the two diagonals, they win. Most games of Tic-Tac-Toe end in a draw, which happens when the board is filled up with neither player having three marks in a row. Instead of a second human player, our artificial intelligence will make moves against the user. The first move to be made is randomly chosen by the program code.

DATA FILE DESIGN

S.No	Field Name	Field Description
1	word_bank	Animals & Birds
2	word_bank	Countries
3	f1	Animals1
4	f2	States
5	f3	Flowers

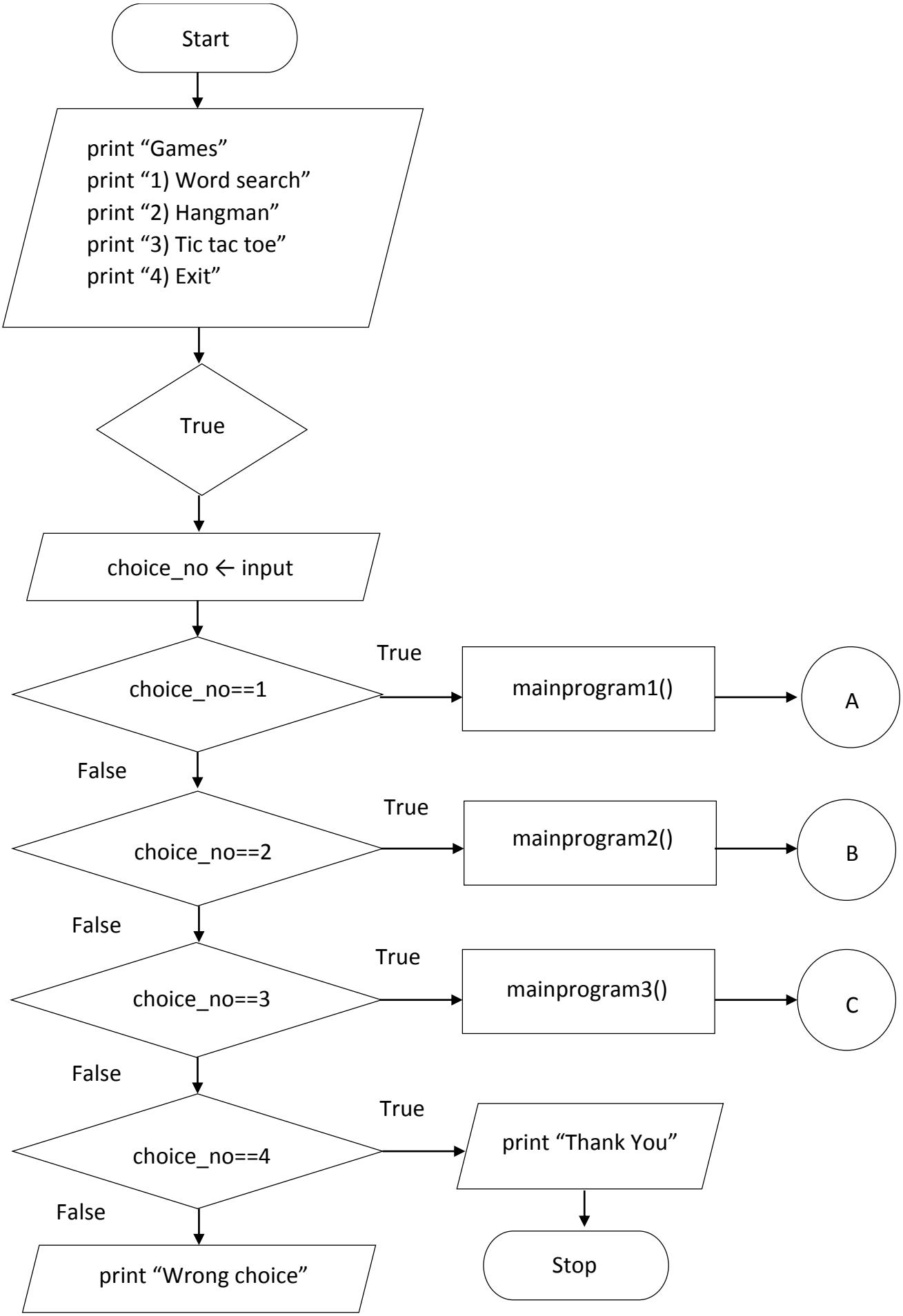
CLASS DESCRIPTION

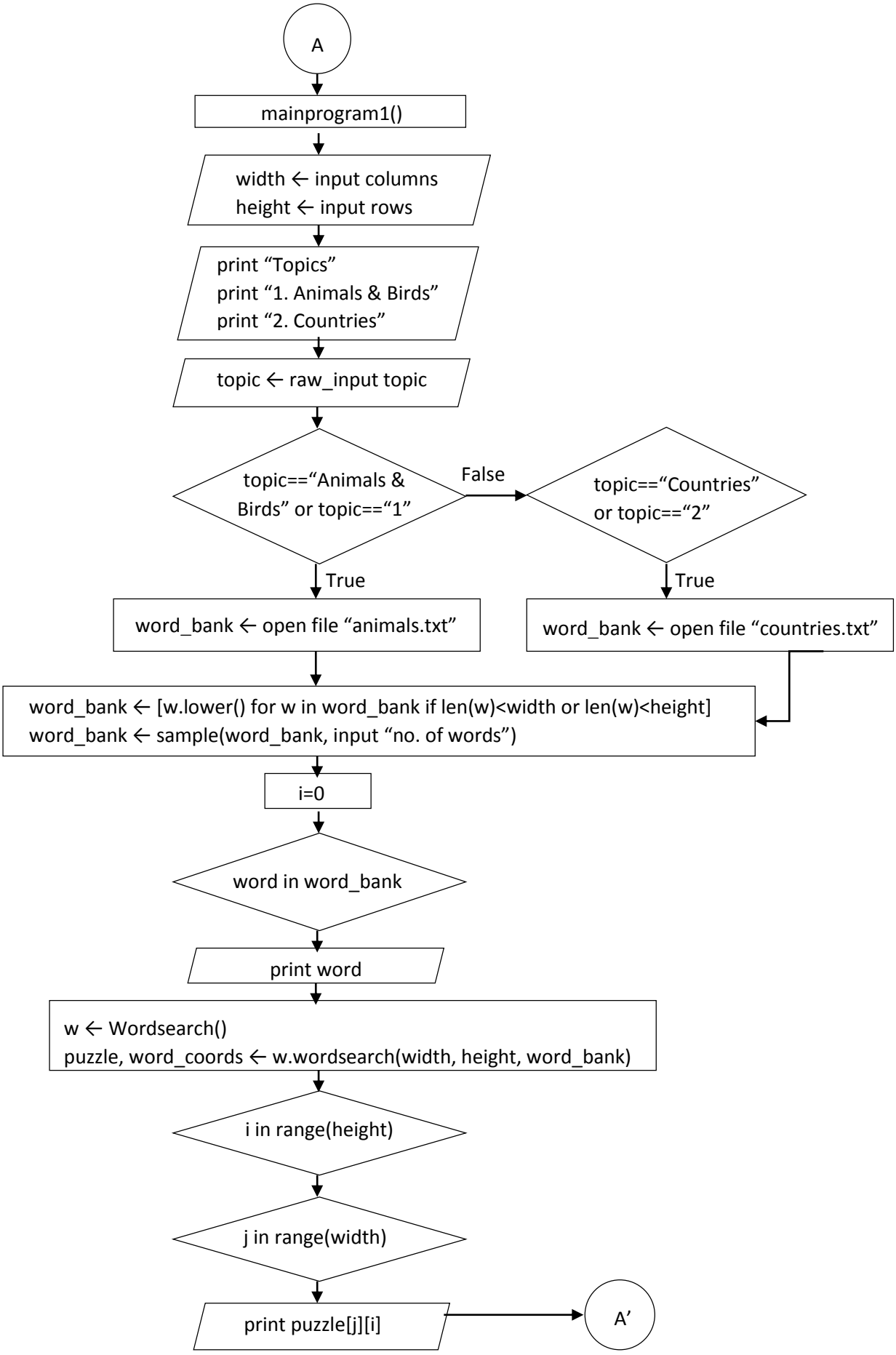
CLASS- WORD SEARCH	
DATA ATTRIBUTES	• width • height • puzzle • words • direction
METHODS	• word_search_base() • insert() • word_search()

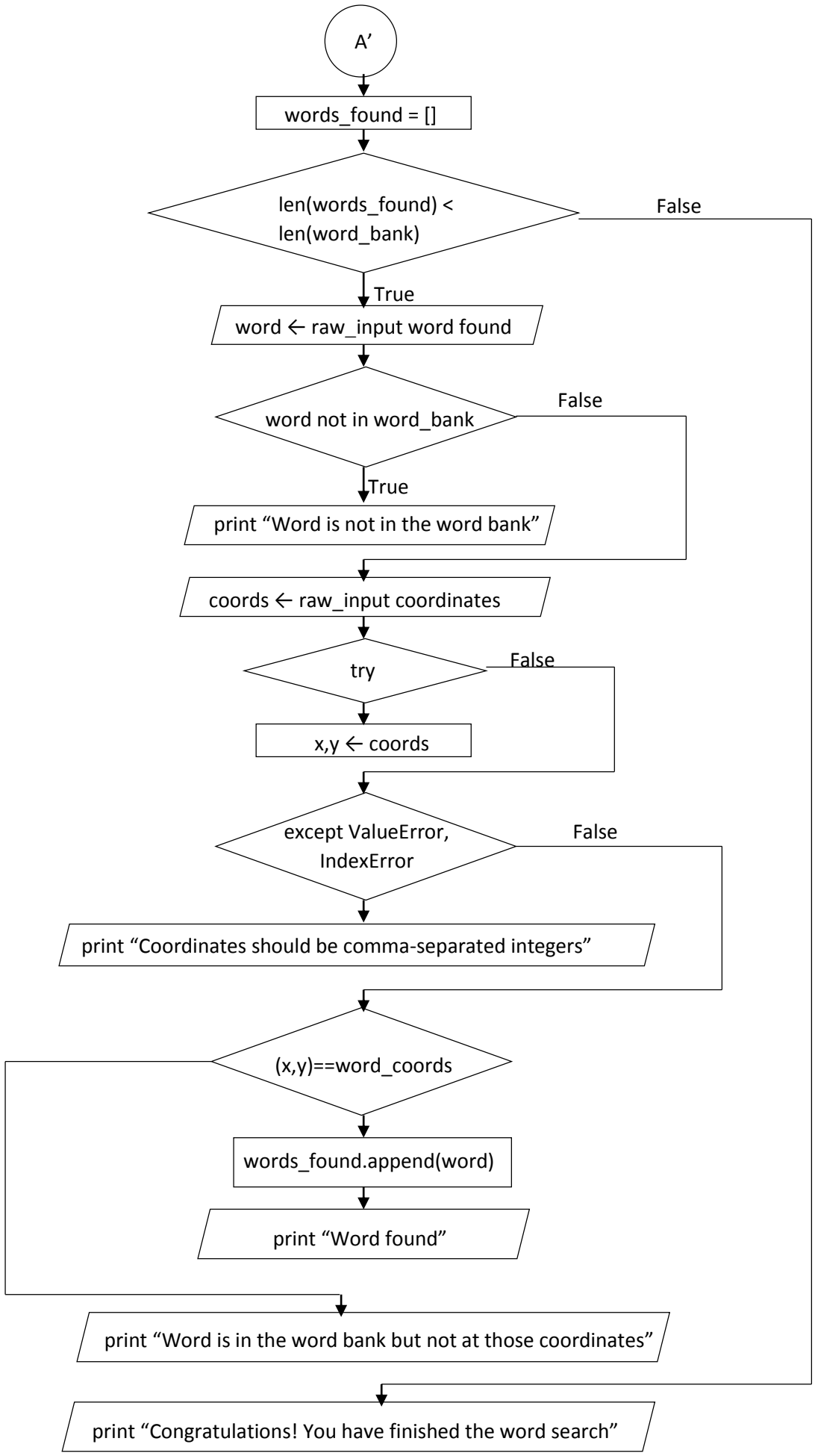
CLASS- HANGMAN	
DATA ATTRIBUTES	• missedLetters • correctLetters • secretWord • alreadyGuessed
METHODS	• getWord() • Board() • getGuess() • playAgain()

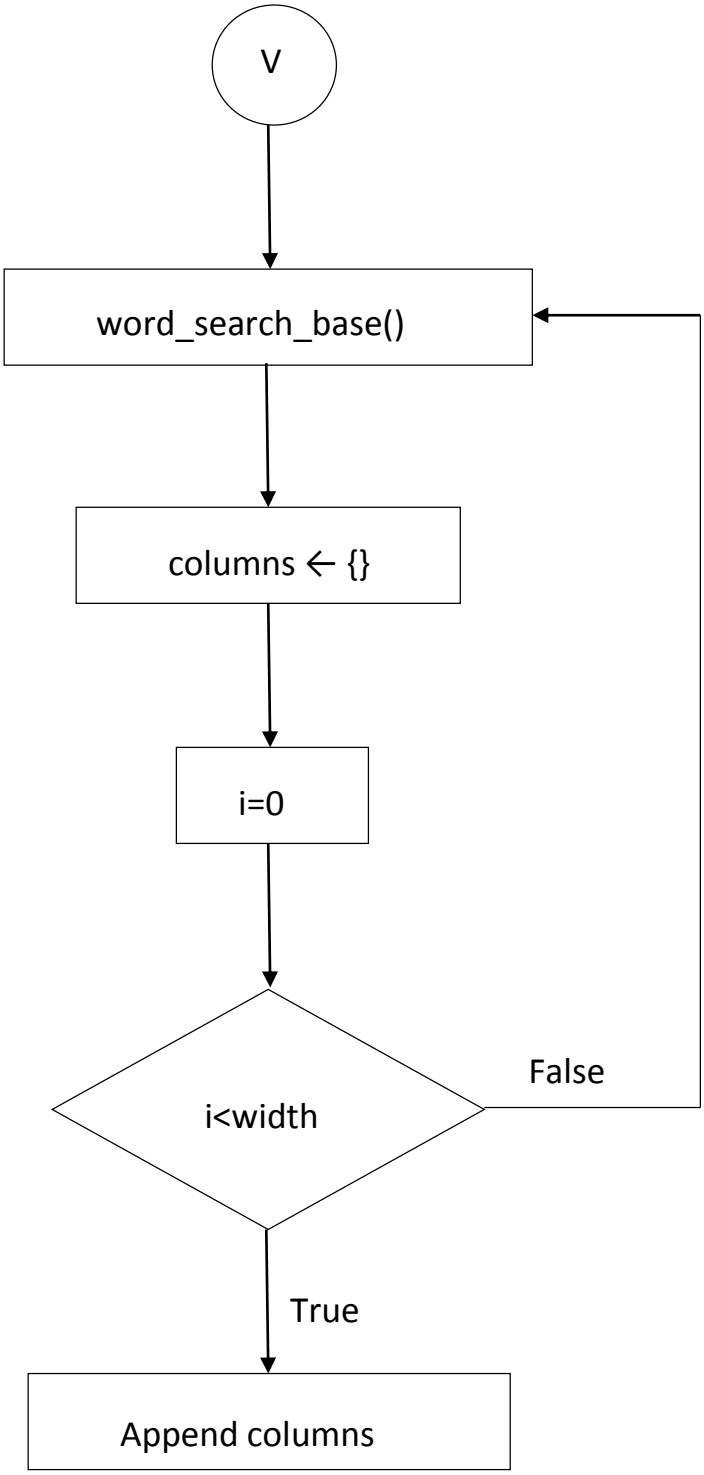
CLASS- TIC-TAC-TOE	
DATA ATTRIBU-TES	• board • letter • move • movesList • ComputerLetter
METHODS	• drawBoard() • inputPlayerLetter() • playfirst() • playAgain() • makeMove() • isWinner() • getBoardCopy() • isSpaceFree() • getPlayerMove() • chooseRandomMovefromlist() • getComputerMove() • isBoardFull()

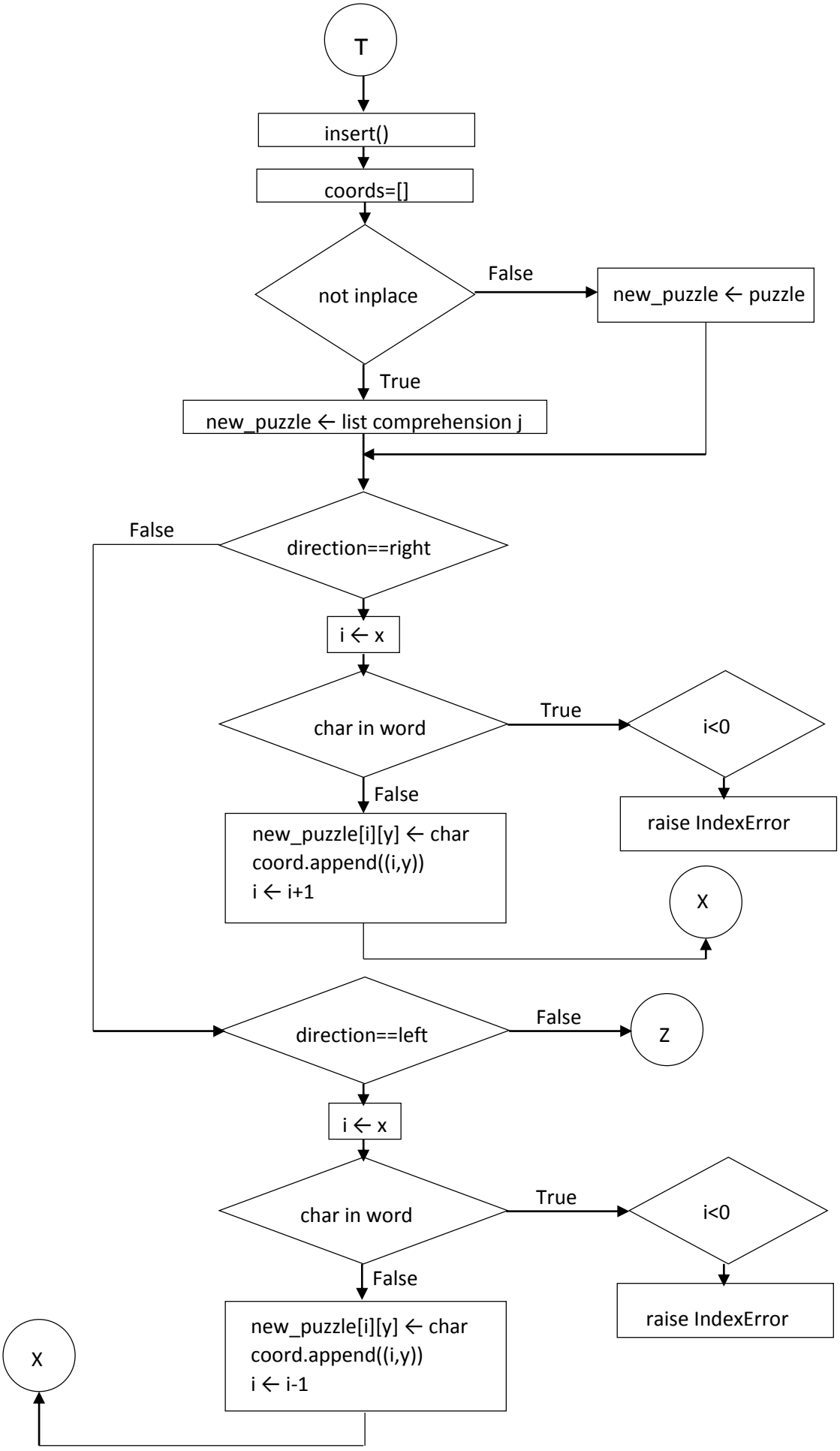
FLOWCHART

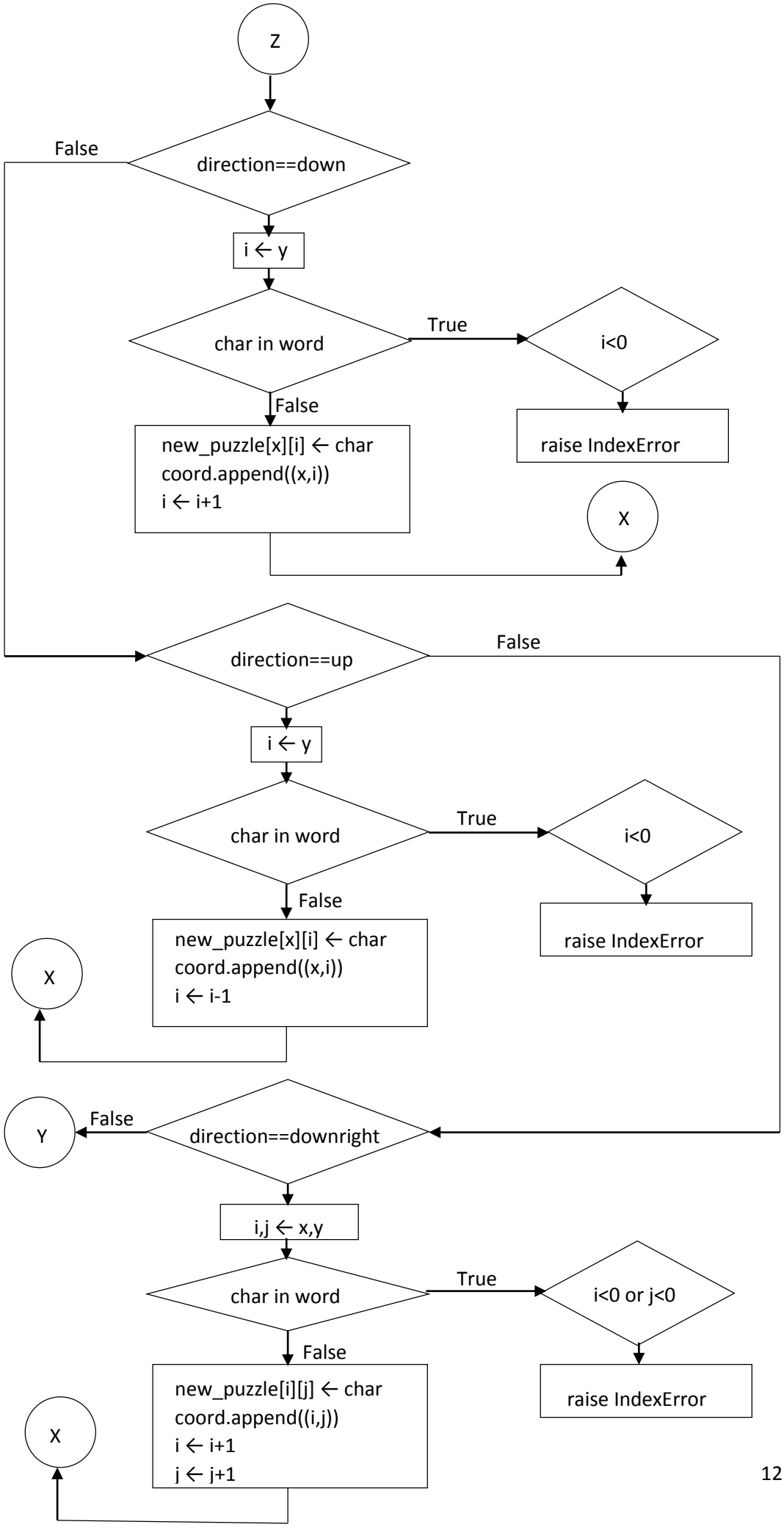

Start
print “Games”
print “1) Word search”
print “2) Hangman”
print “3) Tic tac toe”
print “4) Exit”
True
choice_no ← input
choice_no==1
True
mainprogram1()
A
False
choice_no==2
True
mainprogram2()
B
False
choice_no==3
True
mainprogram3()
C
False
choice_no==4
True
print “Thank You”
Stop
False
print “Wrong choice”

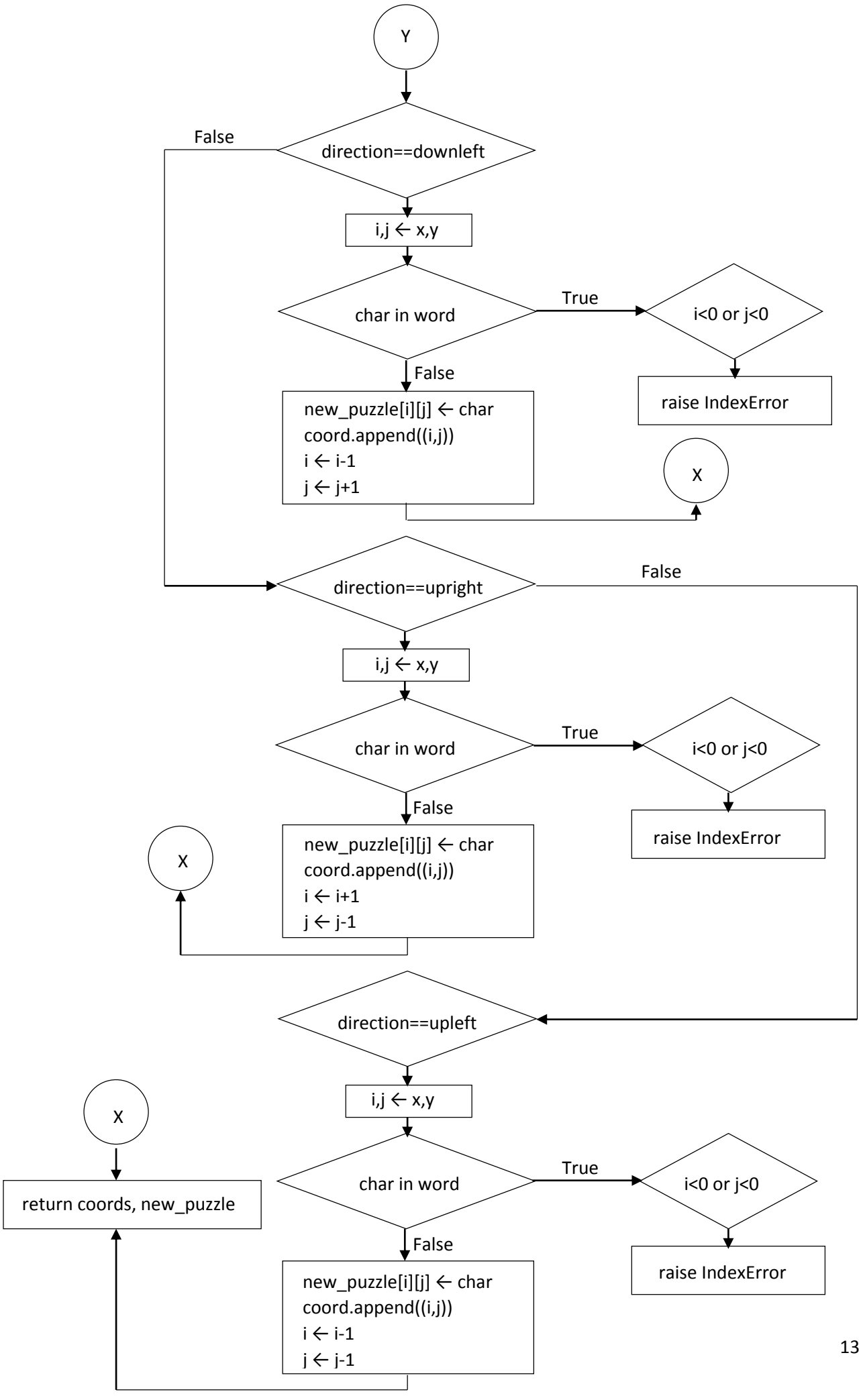

A
mainprogram1()
width ← input columns
height ← input rows
print “Topics”
print “1. Animals & Birds”
print “2. Countries”
topic ← raw_input topic
topic==“Animals & Birds” or topic==“1”
False
topic==“Countries” or topic==“2”
True
True
word_bank ← open file “animals.txt”
word_bank ← open file “countries.txt”
word_bank ← [w.lower() for w in word_bank if len(w)<width or len(w)<height]
word_bank ← sample(word_bank, input “no. of words”)
i=0
word in word_bank
print word
w ← Wordsearch()
puzzle, word_coords ← w.wordsearch(width, height, word_bank)
i in range(height)
j in range(width)
print puzzle[j][i]
A’

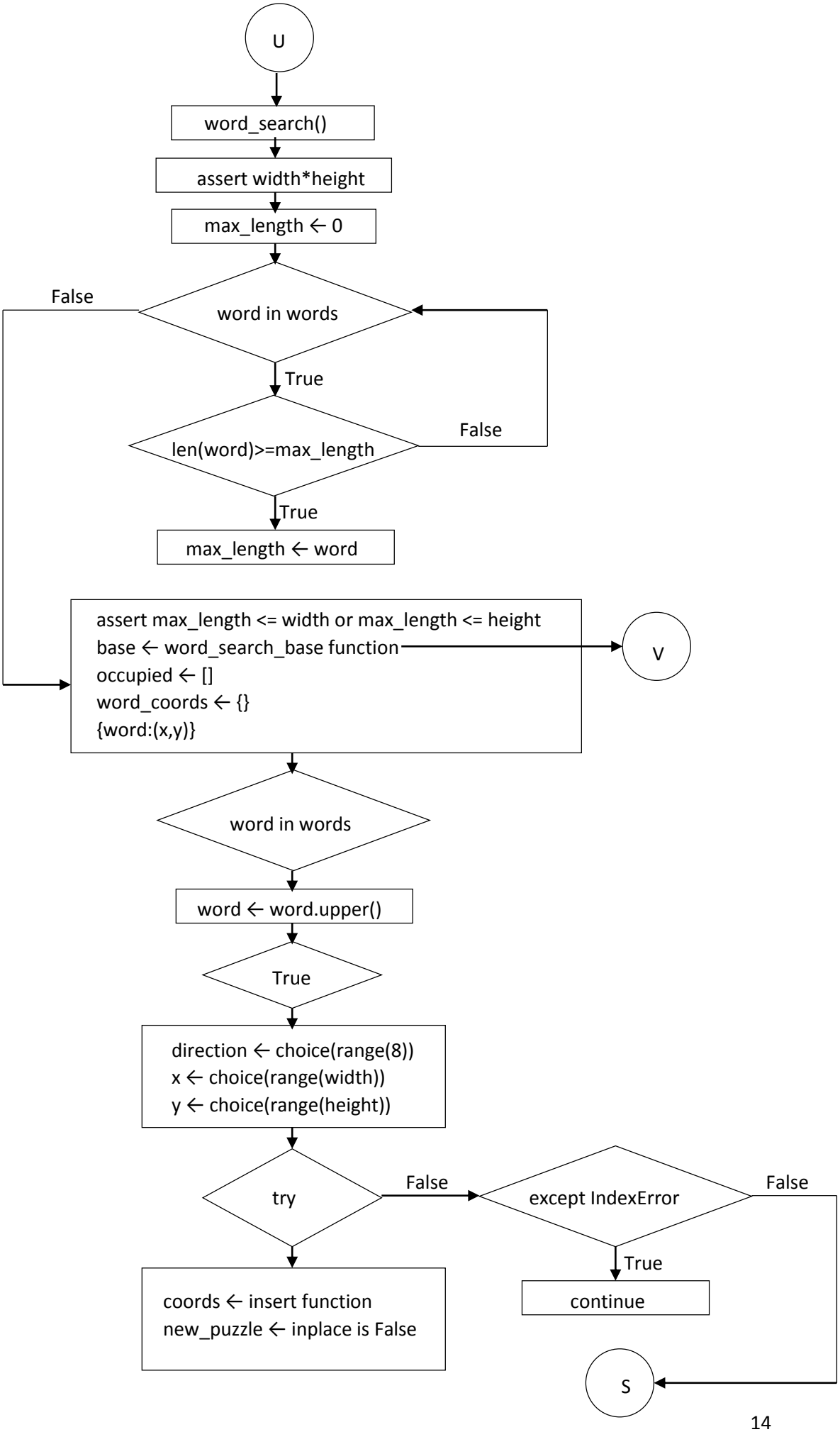

A'
words_found = []
len(words_found) < len(word_bank)
False
True
word ← raw_input word found
word not in word_bank
False
True
print "Word is not in the word bank"
coords ← raw_input coordinates
try
False
x,y ← coords
except ValueError, IndexError
False
print "Coordinates should be comma-separated integers"
(x,y)==word_coords
words_found.append(word)
print "Word found"
print "Word is in the word bank but not at those coordinates"
print "Congratulations! You have finished the word search"

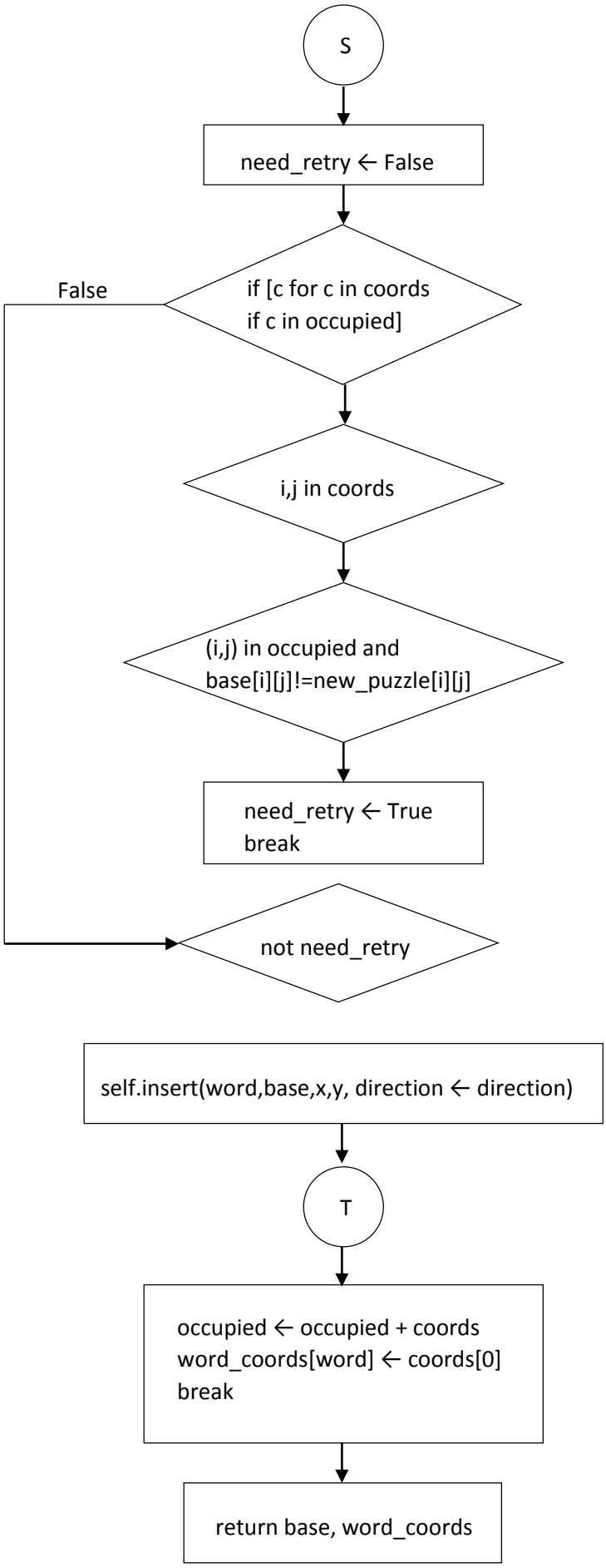

V
word_search_base()
columns ← {}
i=0
i<width
False
True
Append columns

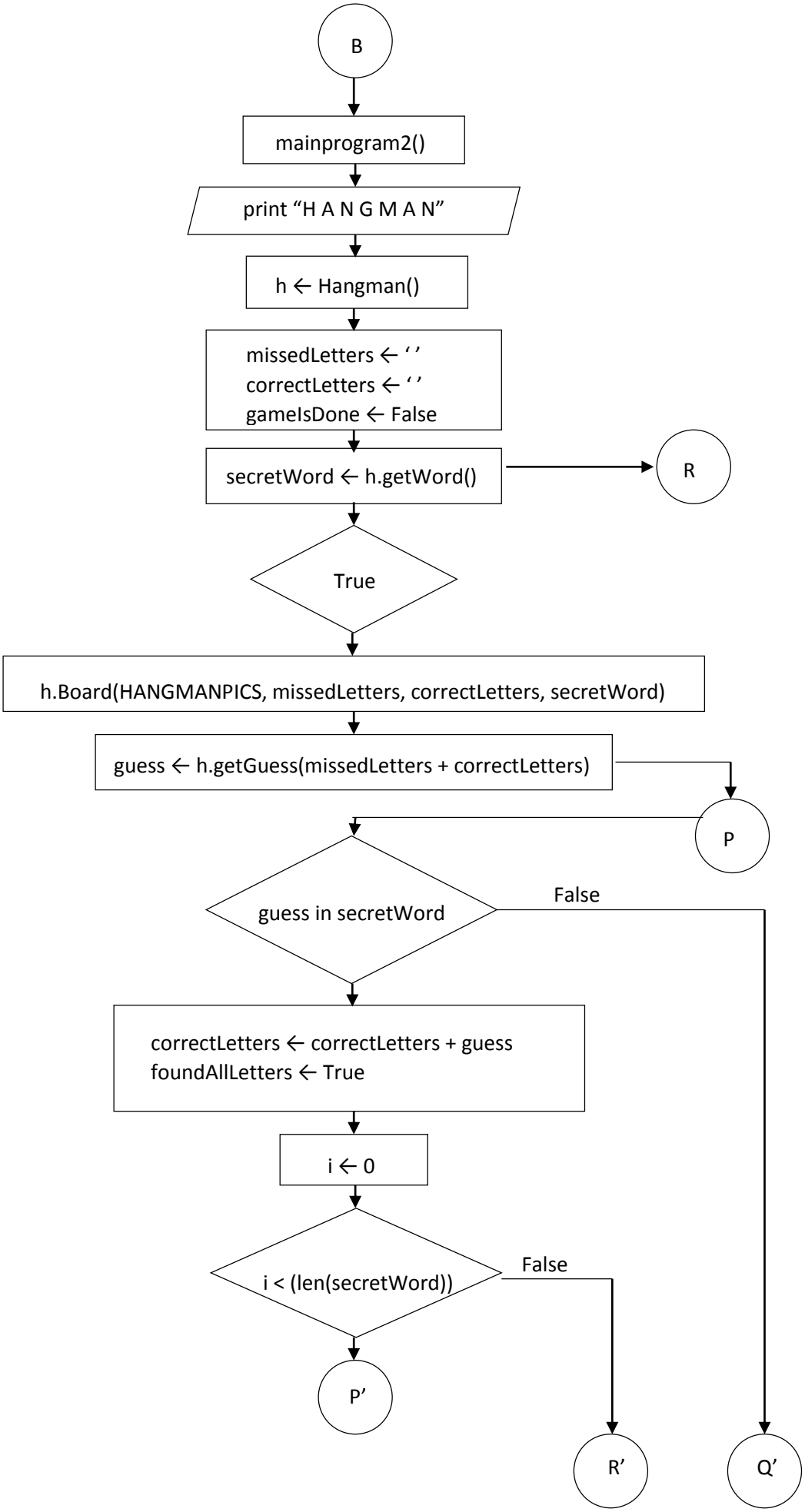

T
insert()
coords=[]
not inplace
False
new_puzzle ← puzzle
True
new_puzzle ← list comprehension j
direction==right
False
i ← x
char in word
True
i<0
raise IndexError
False
new_puzzle[i][y] ← char
coord.append((i,y))
i ← i+1
X
direction==left
False
Z
i ← x
char in word
True
i<0
raise IndexError
False
new_puzzle[i][y] ← char
coord.append((i,y))
i ← i-1
X

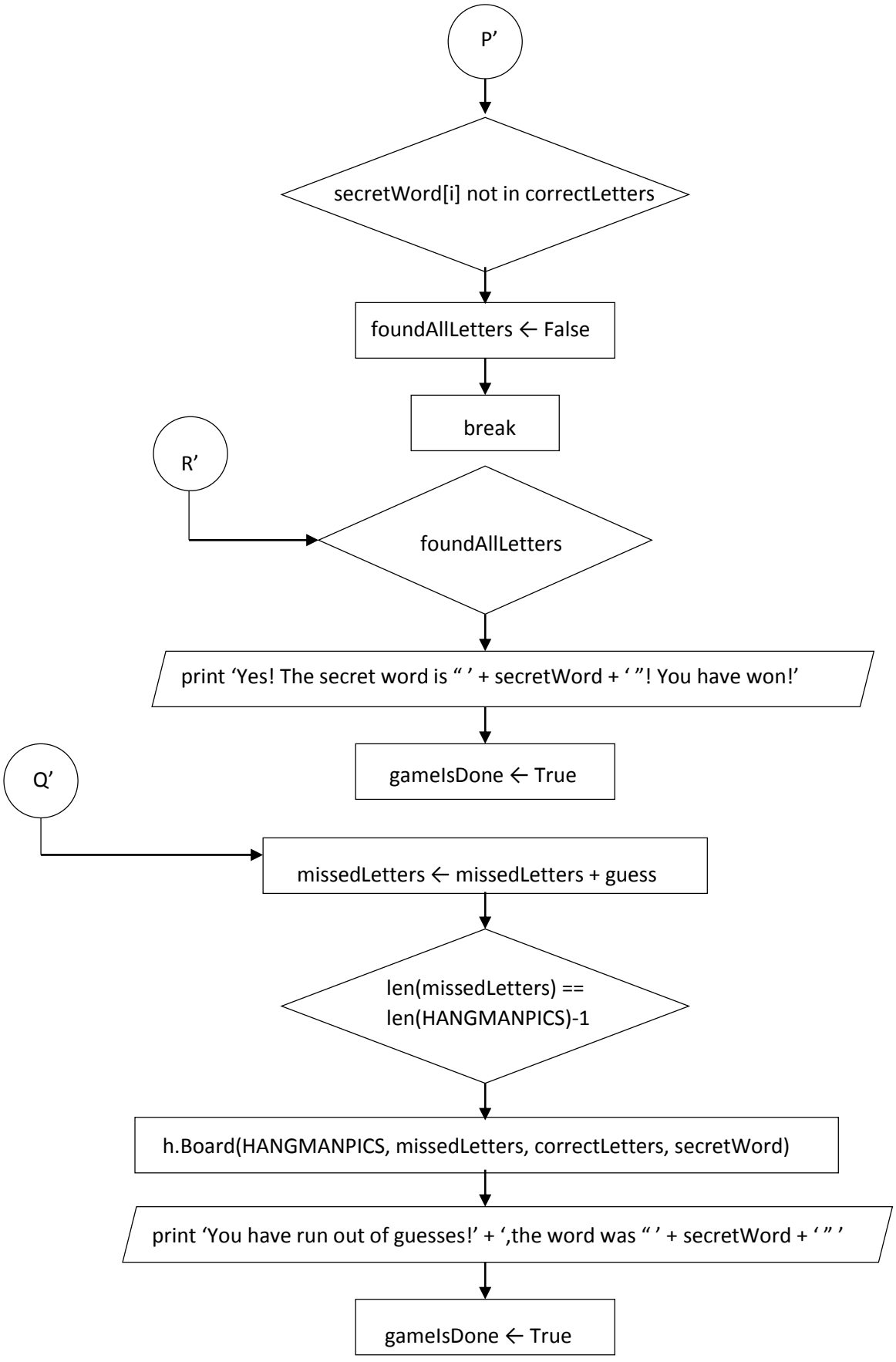

Z
direction==down
False
i ← y
char in word
True
i<0
raise IndexError
False
new_puzzle[x][i] ← char
coord.append((x,i))
i ← i+1
X
direction==up
False
i ← y
char in word
True
i<0
raise IndexError
False
new_puzzle[x][i] ← char
coord.append((x,i))
i ← i-1
X
Y
False
direction==downright
i,j ← x,y
char in word
True
i<0 or j<0
raise IndexError
False
new_puzzle[i][j] ← char
coord.append((i,j))
i ← i+1
j ← j+1
X

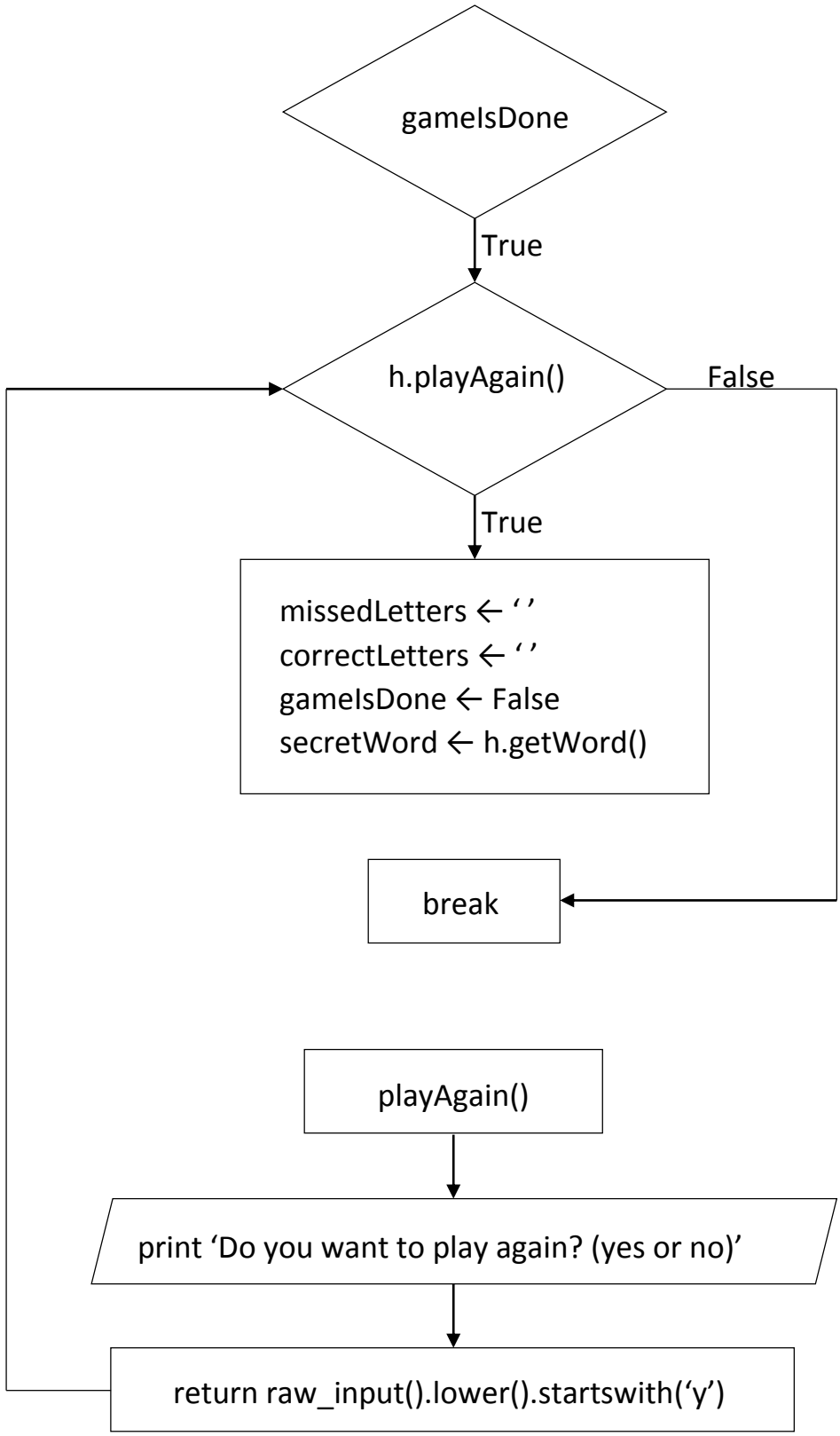

Y
direction==downleft
False
i,j ← x,y
char in word
True
i<0 or j<0
raise IndexError
False
new_puzzle[i][j] ← char
coord.append((i,j))
i ← i-1
j ← j+1
X
direction==upright
False
i,j ← x,y
char in word
True
i<0 or j<0
raise IndexError
False
X
new_puzzle[i][j] ← char
coord.append((i,j))
i ← i+1
j ← j-1
direction==upleft
X
i,j ← x,y
char in word
True
i<0 or j<0
raise IndexError
return coords, new_puzzle
False
new_puzzle[i][j] ← char
coord.append((i,j))
i ← i-1
j ← j-1

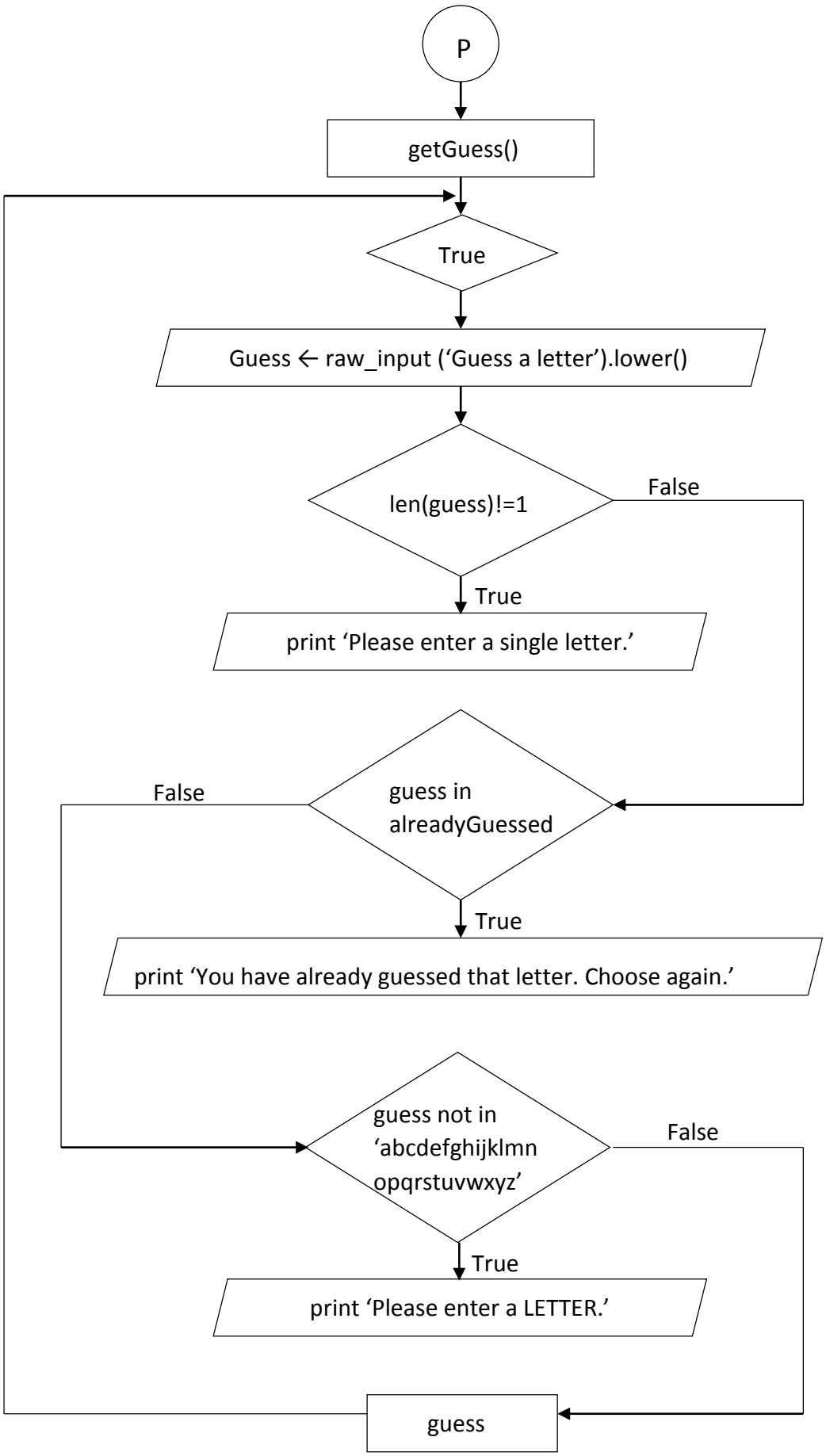

U
word_search()
assert width*height
max_length ← 0
False
word in words
True
len(word)>=max_length
False
True
max_length ← word
assert max_length <= width or max_length <= height
base ← word_search_base function
occupied ← []
word_coords ← {}
{word:(x,y)}
V
word in words
word ← word.upper()
True
direction ← choice(range(8))
x ← choice(range(width))
y ← choice(range(height))
try
False
except IndexError
False
True
coords ← insert function
new_puzzle ← inplace is False
continue
S

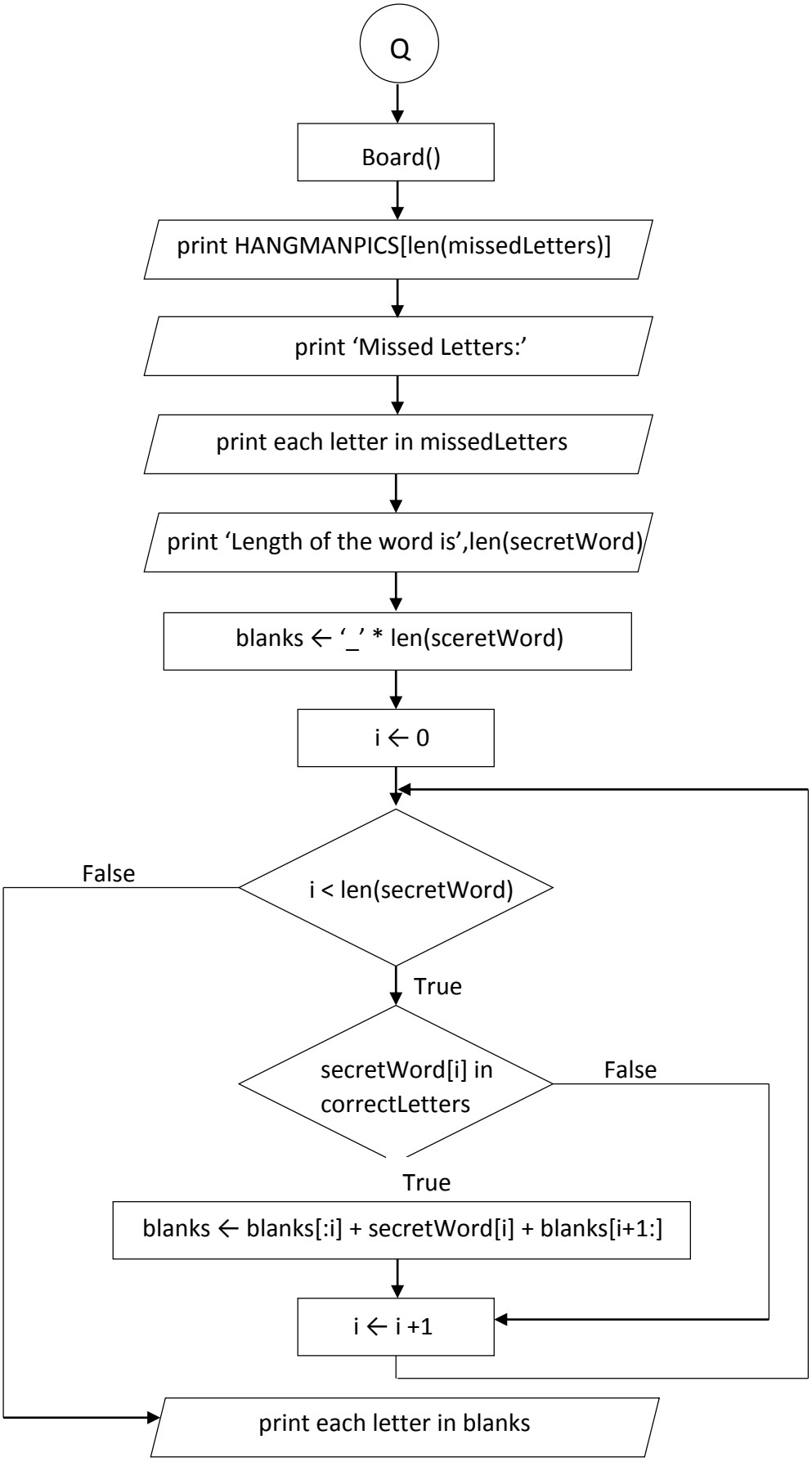

S
need_retry ← False
False
if [c for c in coords
if c in occupied]
i,j in coords
(i,j) in occupied and
base[i][j]!=new_puzzle[i][j]
need_retry ← True
break
not need_retry
self.insert(word,base,x,y, direction ← direction)
T
occupied ← occupied + coords
word_coords[word] ← coords[0]
break
return base, word_coords

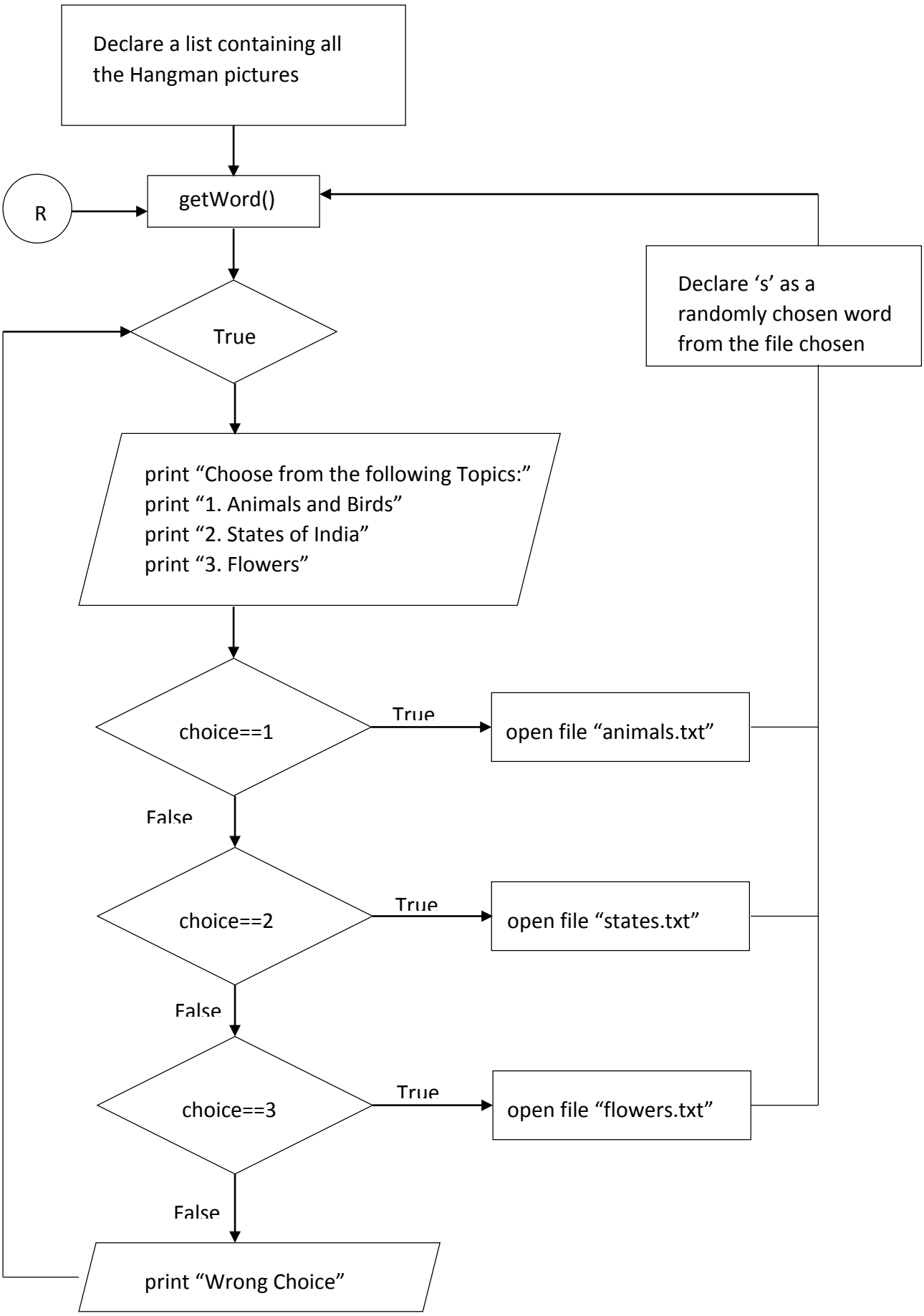

B
mainprogram2()
print “H A N G M A N”
h ← Hangman()
missedLetters ← ‘ ’
correctLetters ← ‘ ’
gameIsDone ← False
secretWord ← h.getWord()
R
True
h.Board(HANGMANPICS, missedLetters, correctLetters, secretWord)
guess ← h.getGuess(missedLetters + correctLetters)
P
guess in secretWord
False
correctLetters ← correctLetters + guess
foundAllLetters ← True
i ← 0
i < (len(secretWord))
False
P’
R’
Q’

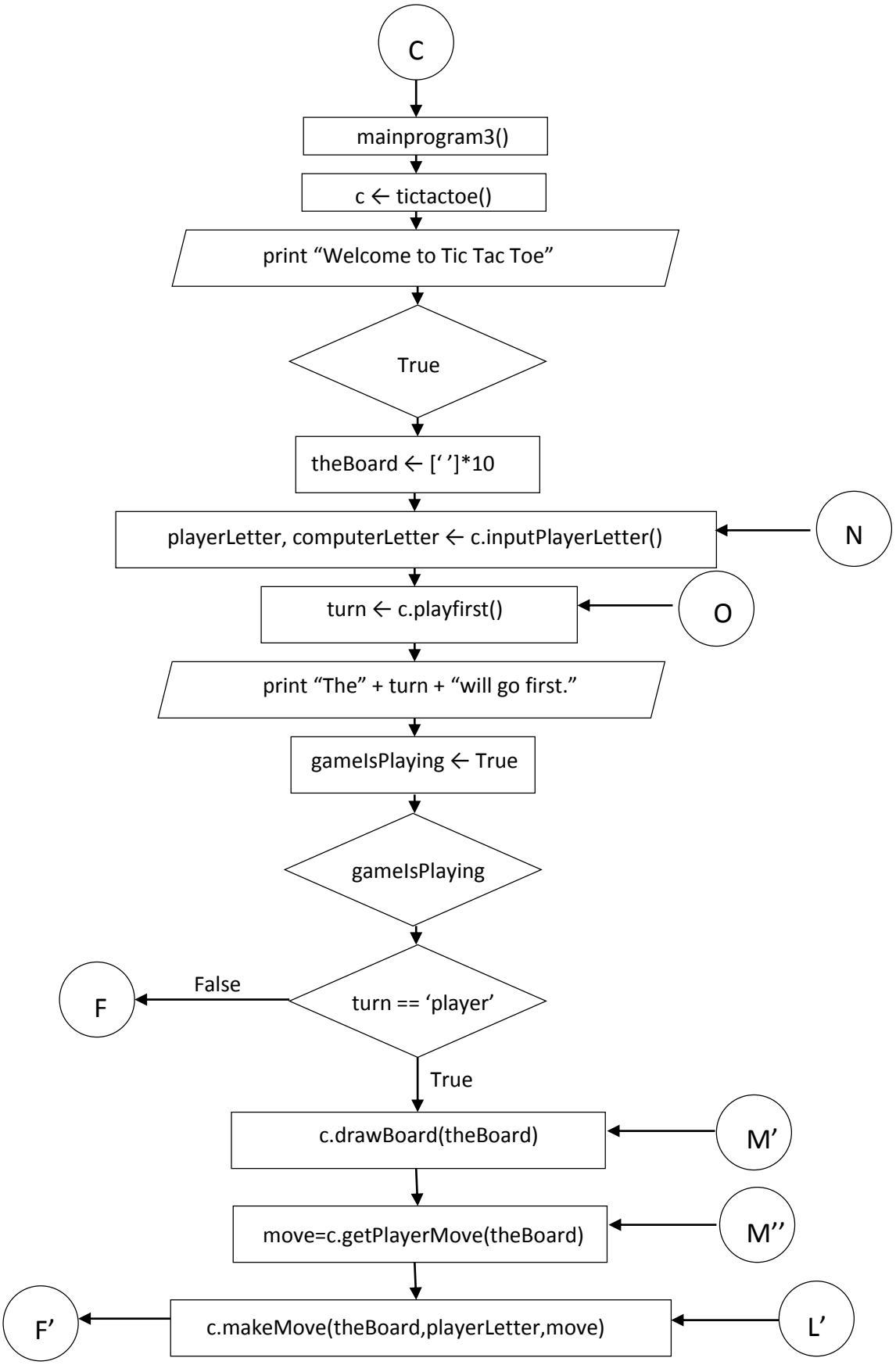

P’
secretWord[i] not in correctLetters
foundAllLetters ← False
break
R’
foundAllLetters
print ‘Yes! The secret word is “ ’ + secretWord + ‘ ”! You have won!’
gameIsDone ← True
Q’
missedLetters ← missedLetters + guess
len(missedLetters) ==
len(HANGMANPICS)-1
h.Board(HANGMANPICS, missedLetters, correctLetters, secretWord)
print ‘You have run out of guesses!’ + ‘,the word was “ ’ + secretWord + ‘ ” ’
gameIsDone ← True

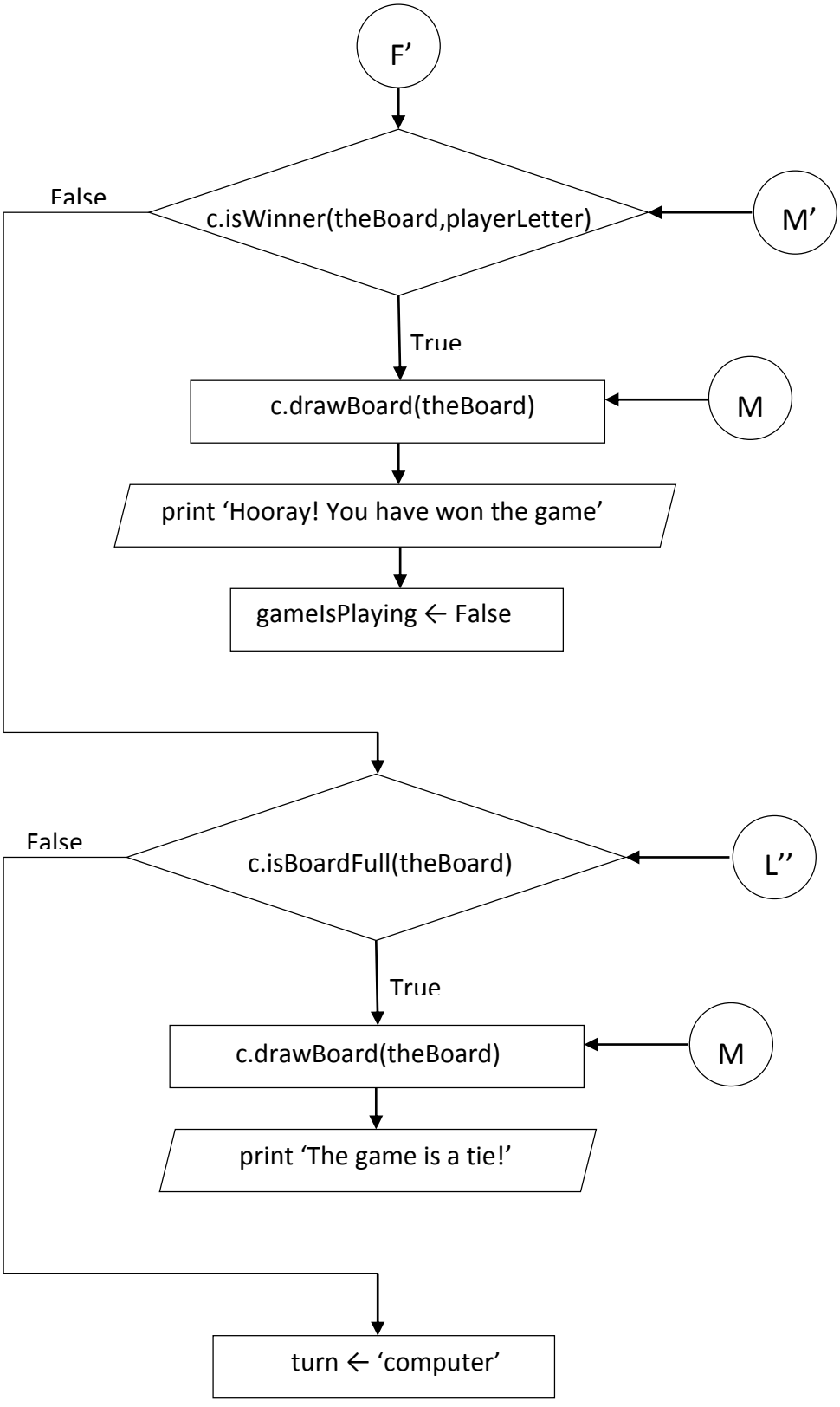

gameIsDone
True
h.playAgain()
False
True
missedLetters ← ‘ ’
correctLetters ← ‘ ’
gameIsDone ← False
secretWord ← h.getWord()
break
playAgain()
print ‘Do you want to play again? (yes or no)’
return raw_input().lower().startswith(‘y’)

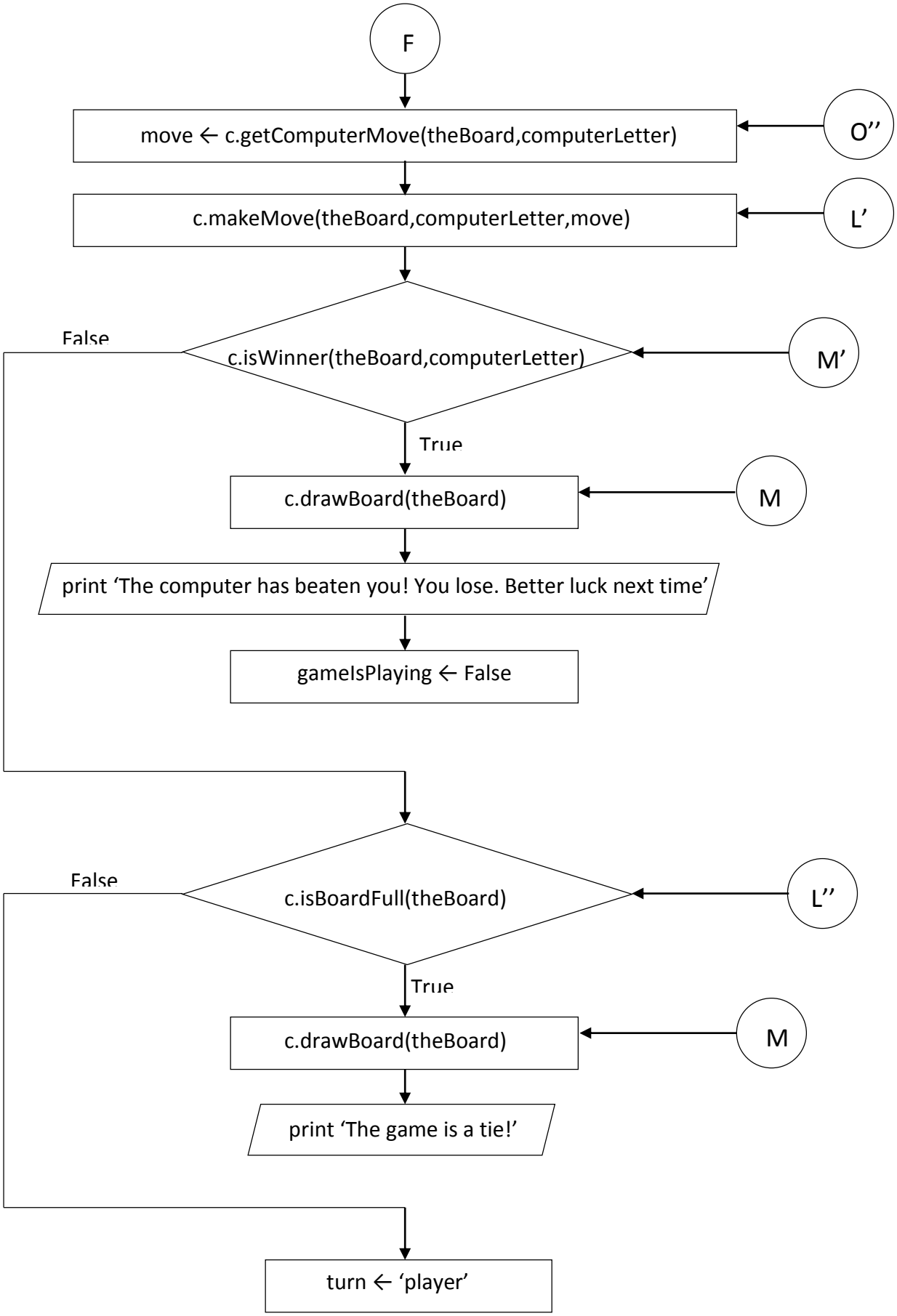

P
getGuess()
True
Guess ← raw_input ('Guess a letter').lower()
len(guess)!=1
False
True
print 'Please enter a single letter.'
guess in alreadyGuessed
False
True
print 'You have already guessed that letter. Choose again.'
guess not in 'abcdefghijklmn opqrstuvwxyz'
False
True
print 'Please enter a LETTER.'
guess

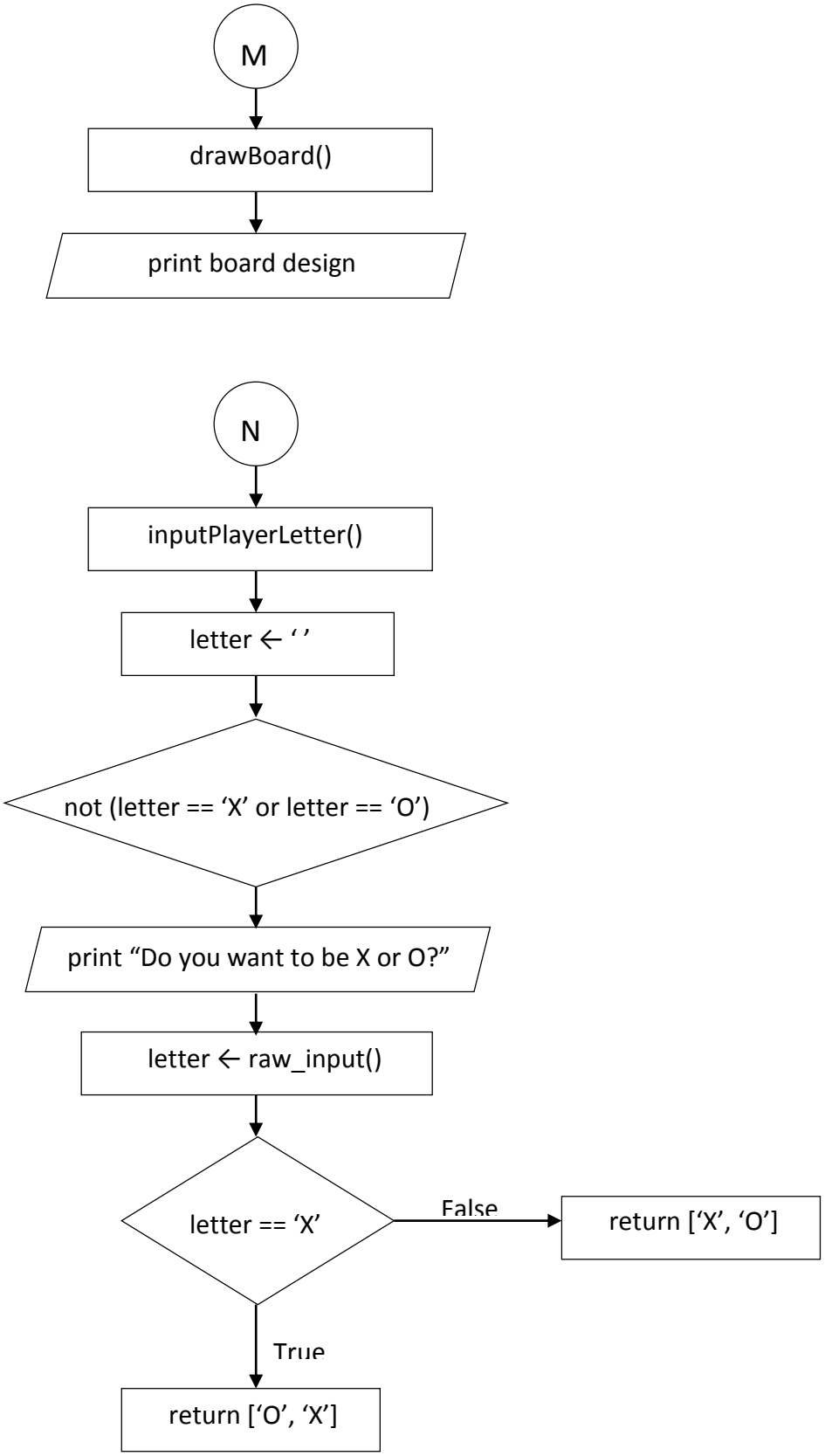

Q
Board()
print HANGMANPICS[len(missedLetters)]
print 'Missed Letters:'
print each letter in missedLetters
print 'Length of the word is',len(secretWord)
blanks ← '_' * len(sceretWord)
i ← 0
i < len(secretWord)
False
True
secretWord[i] in correctLetters
False
True
blanks ← blanks[:i] + secretWord[i] + blanks[i+1:]
i ← i +1
print each letter in blanks

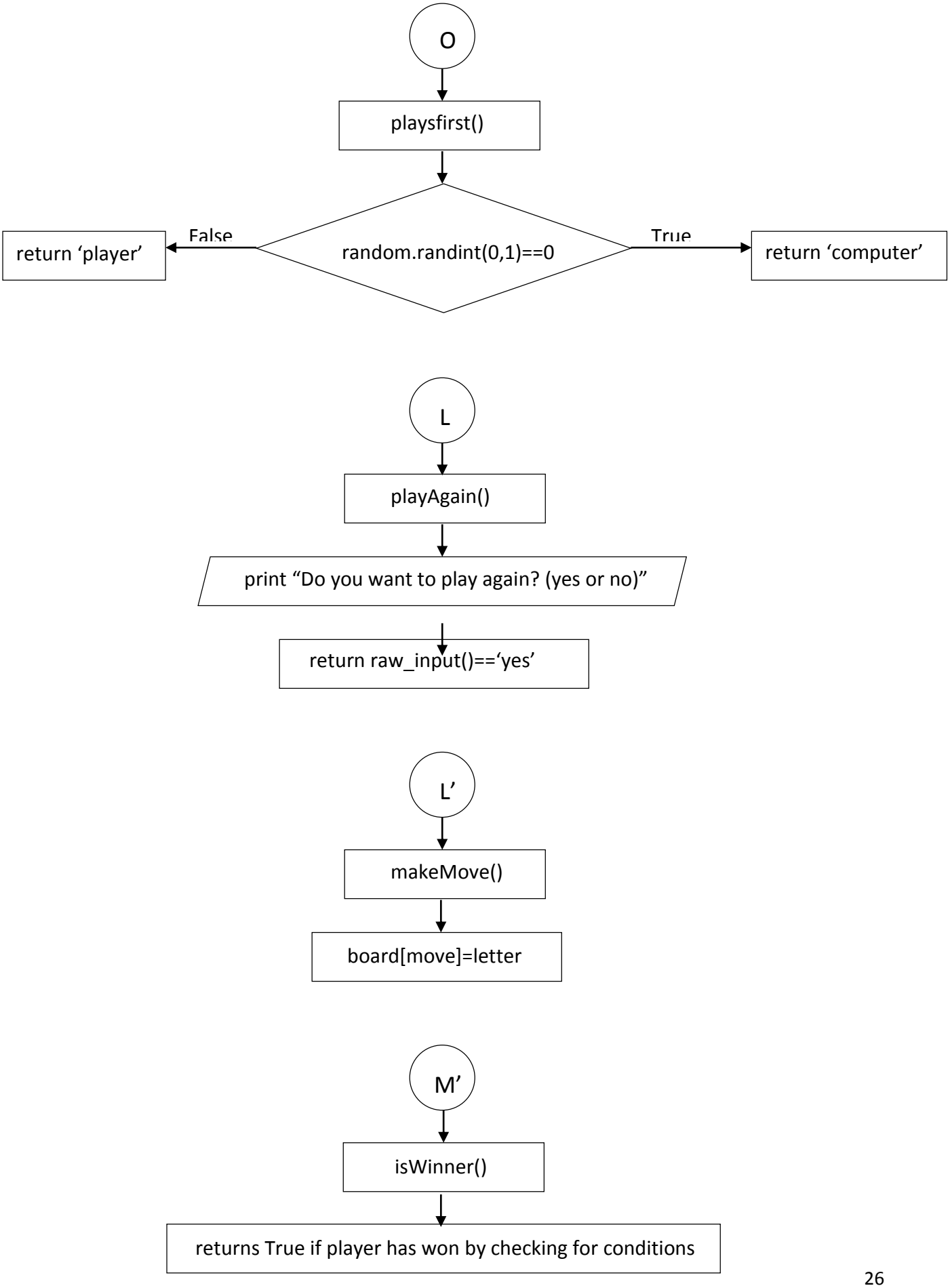

Declare a list containing all the Hangman pictures
R
getWord()
Declare 's' as a randomly chosen word from the file chosen
True
print "Choose from the following Topics:"
print "1. Animals and Birds"
print "2. States of India"
print "3. Flowers"
choice==1
True
open file "animals.txt"
False
choice==2
True
open file "states.txt"
False
choice==3
True
open file "flowers.txt"
False
print "Wrong Choice"

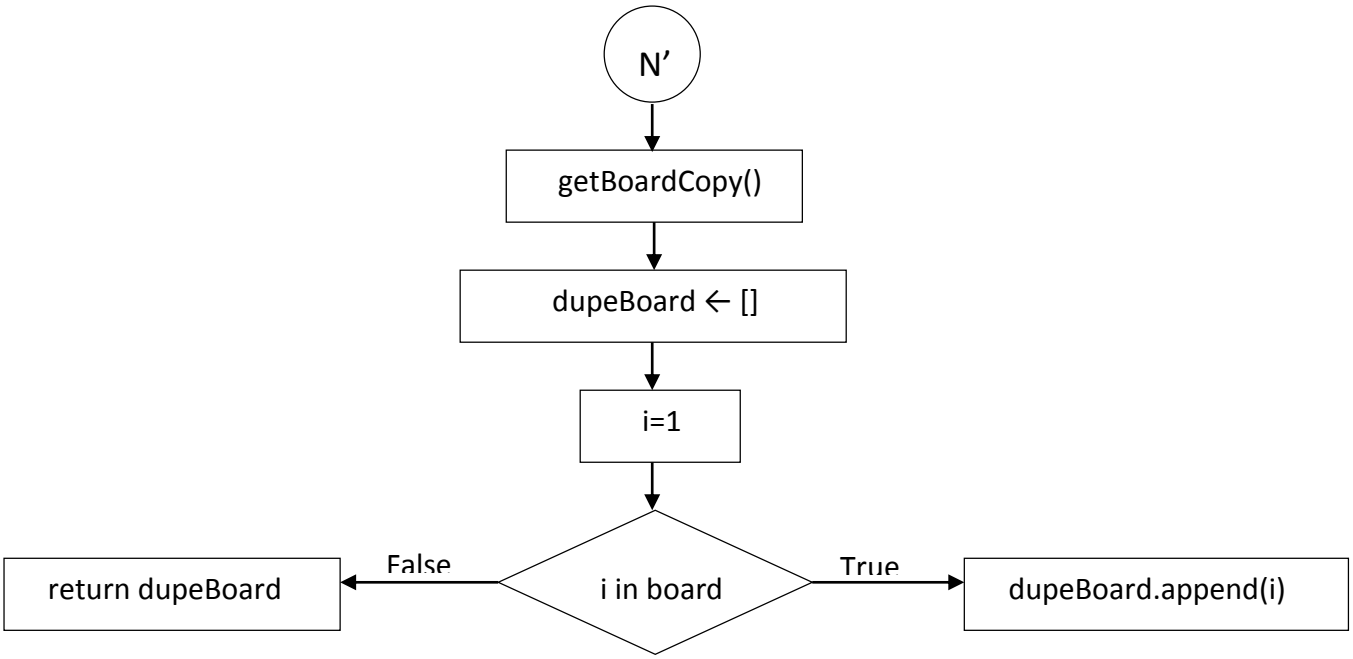

C
mainprogram3()
c ← tictactoe()
print “Welcome to Tic Tac Toe”
True
theBoard ← [‘ ’]*10
playerLetter, computerLetter ← c.inputPlayerLetter()
N
turn ← c.playfirst()
O
print “The” + turn + “will go first.”
gameIsPlaying ← True
gameIsPlaying
F
False
turn == ‘player’
True
c.drawBoard(theBoard)
M’
move=c.getPlayerMove(theBoard)
M’’
F’
c.makeMove(theBoard,playerLetter,move)
L’

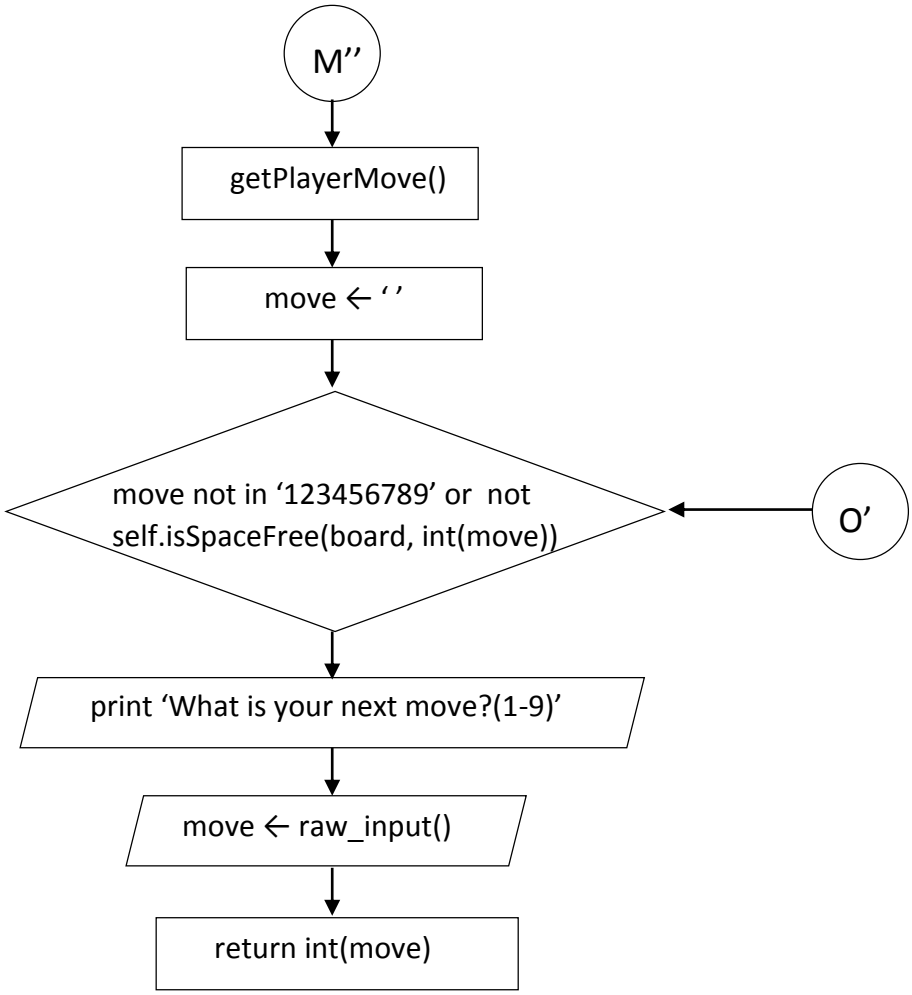

F'
c.isWinner(theBoard,playerLetter)
False
M'
True
c.drawBoard(theBoard)
M
print 'Hooray! You have won the game'
gameIsPlaying ← False
c.isBoardFull(theBoard)
False
L''
True
c.drawBoard(theBoard)
M
print 'The game is a tie!'
turn ← 'computer'

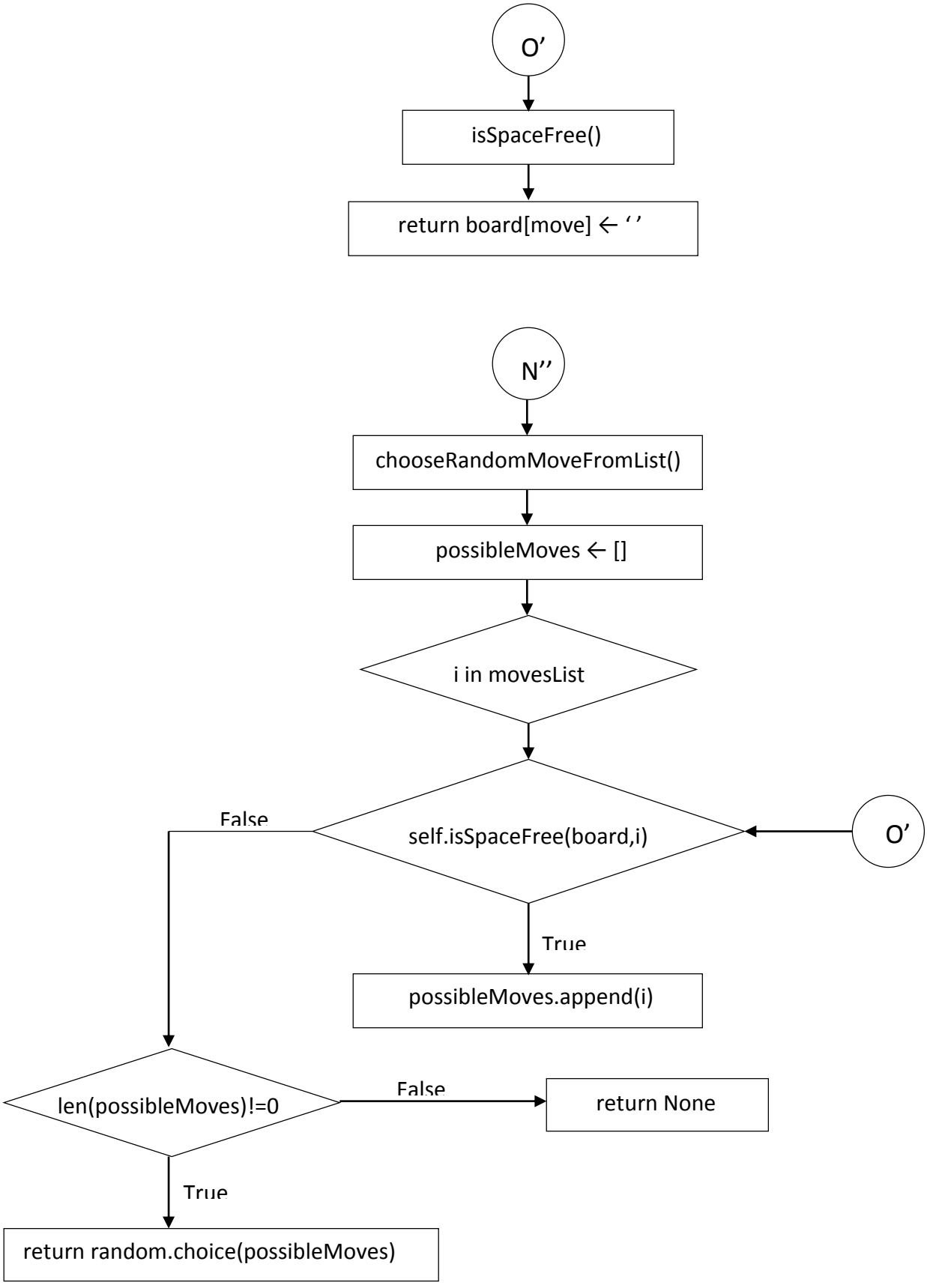

F
move ← c.getComputerMove(theBoard,computerLetter)
O''
c.makeMove(theBoard,computerLetter,move)
L'
False
c.isWinner(theBoard,computerLetter)
M'
True
c.drawBoard(theBoard)
M
print 'The computer has beaten you! You lose. Better luck next time'
gameIsPlaying ← False
False
c.isBoardFull(theBoard)
L''
True
c.drawBoard(theBoard)
M
print 'The game is a tie!'
turn ← 'player'

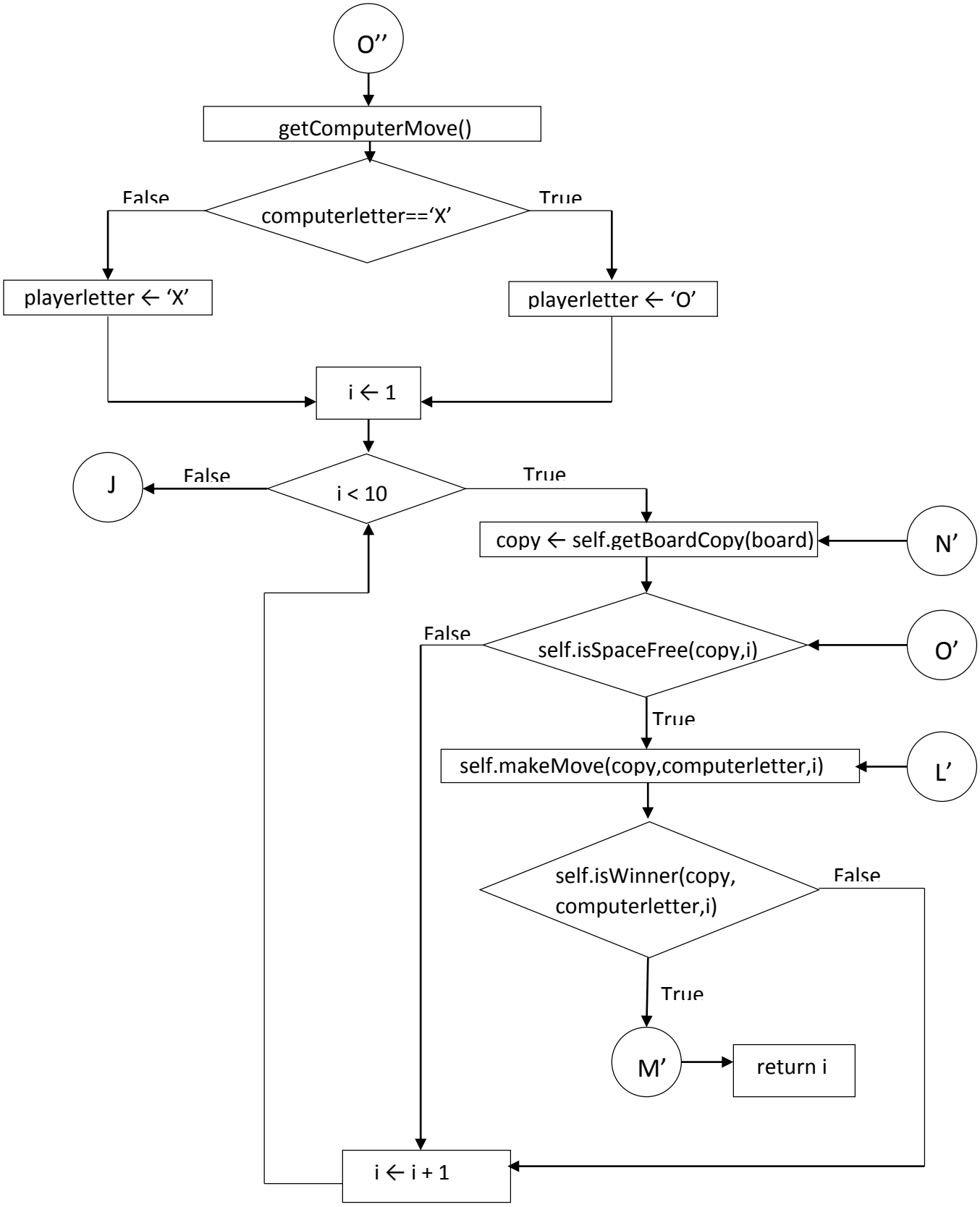

M
drawBoard()
print board design
N
inputPlayerLetter()
letter ← ‘ ’
not (letter == ‘X’ or letter == ‘O’)
print “Do you want to be X or O?”
letter ← raw_input()
letter == ‘X’
False
return [‘X’, ‘O’]
True
return [‘O’, ‘X’]

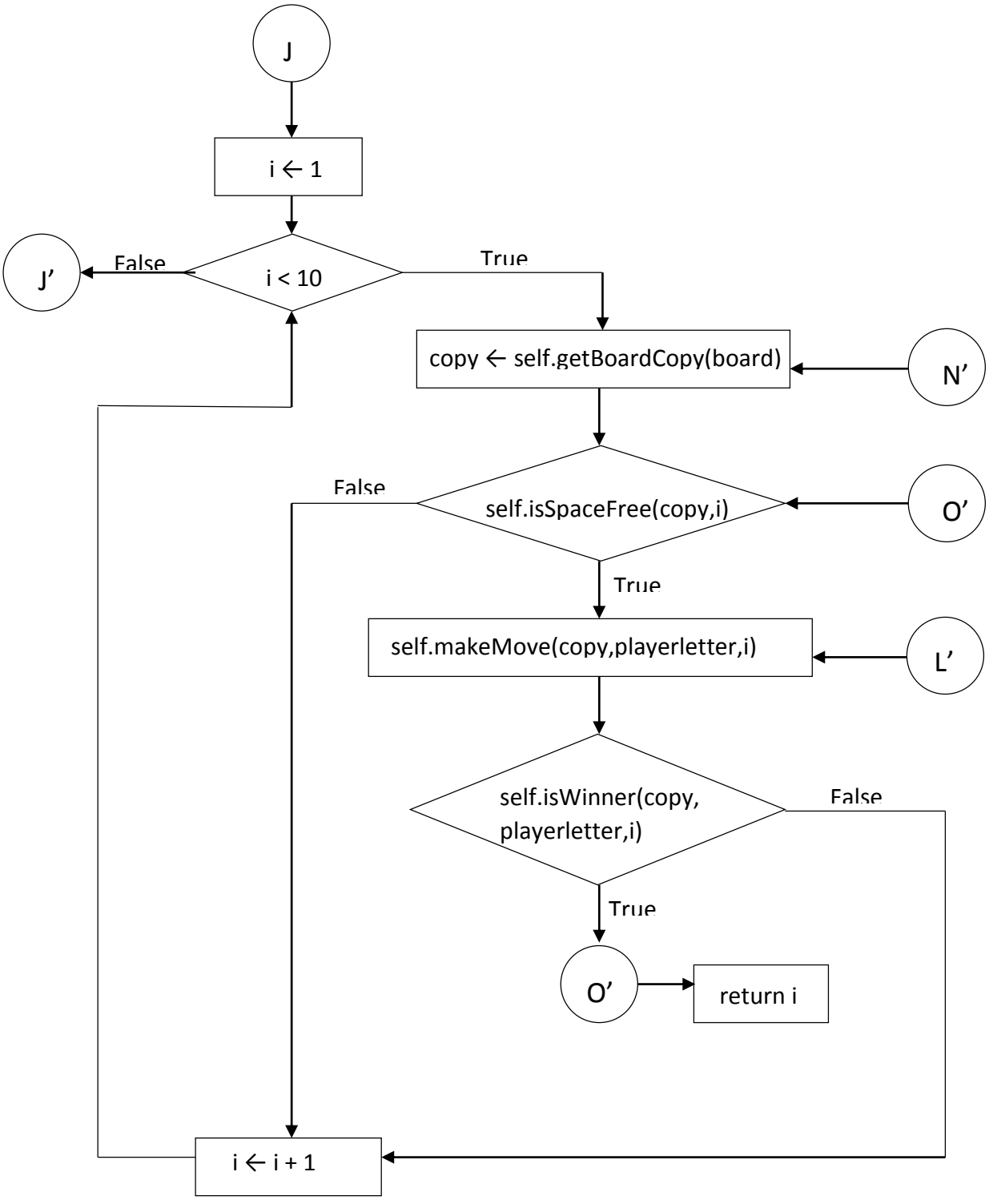

O
playsfirst()
random.randint(0,1)==0
False
return 'player'
True
return 'computer'
L
playAgain()
print "Do you want to play again? (yes or no)"
return raw_input()=='yes'
L'
makeMove()
board[move]=letter
M'
isWinner()
returns True if player has won by checking for conditions

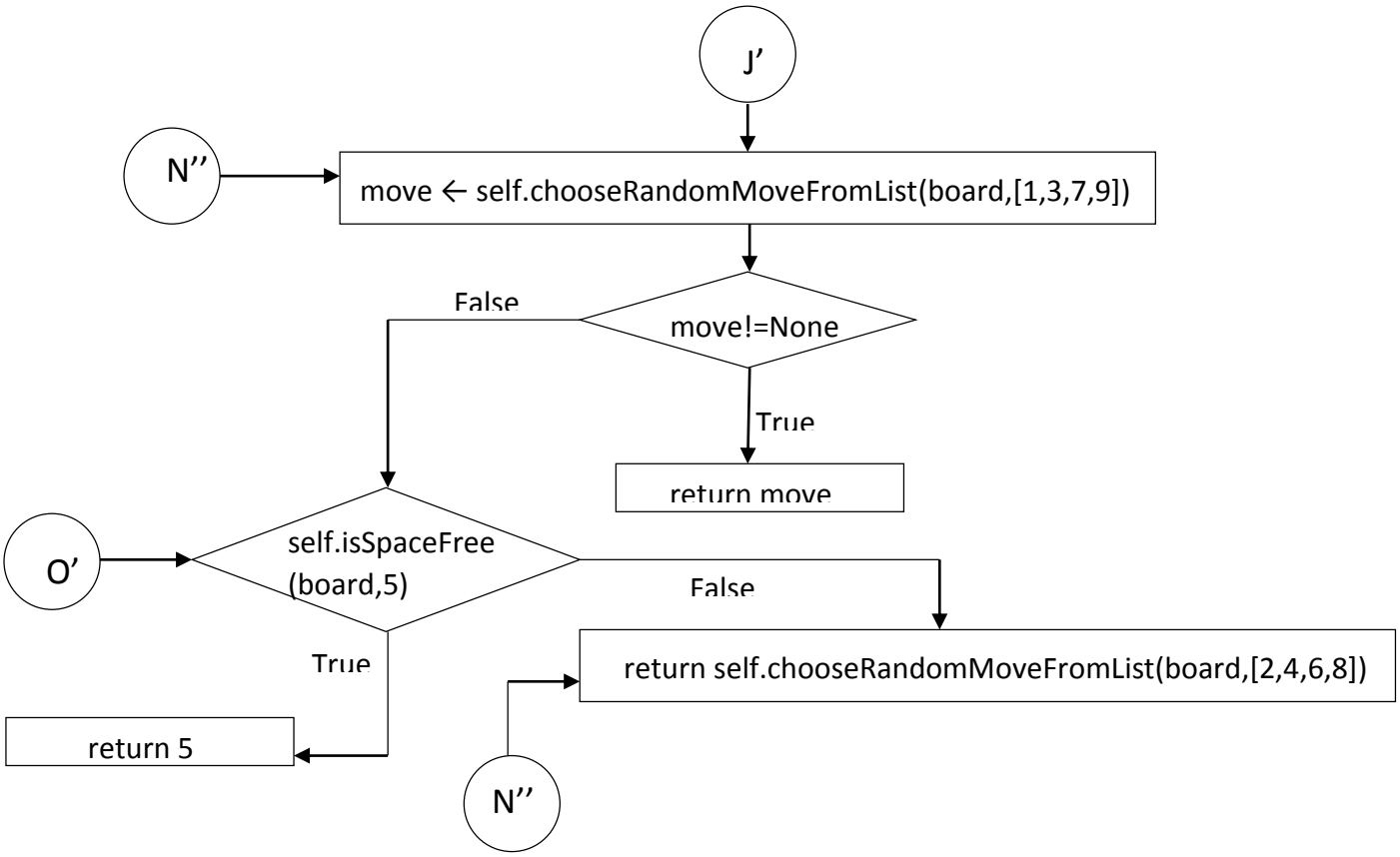

N’
getBoardCopy()
dupeBoard ← []
i=1
i in board
False
True
return dupeBoard
dupeBoard.append(i)

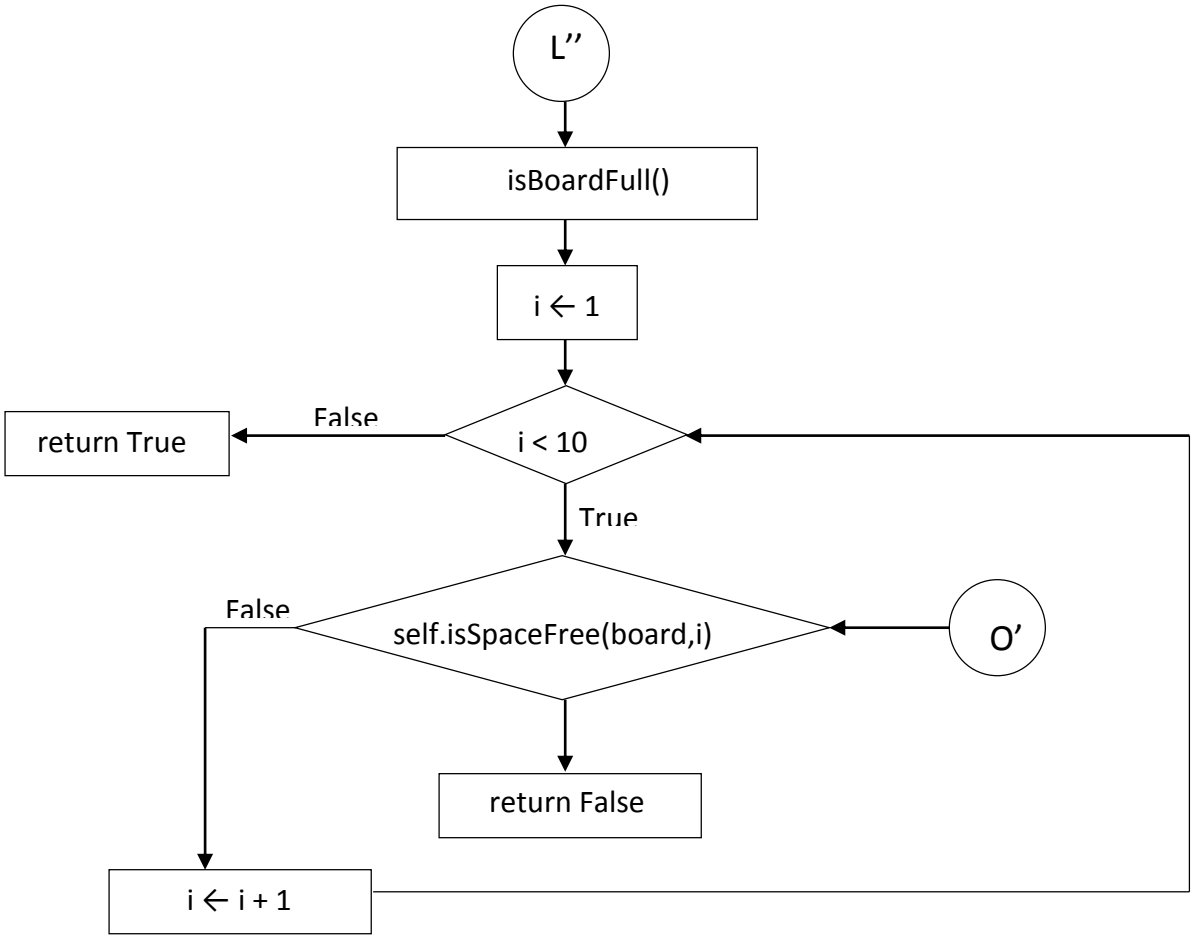

M’’
getPlayerMove()
move ← ‘ ’
move not in ‘123456789’ or not
self.isSpaceFree(board, int(move))
O’
print ‘What is your next move?(1-9)’
move ← raw_input()
return int(move)


O’
isSpaceFree()
return board[move] ← ‘ ’
N’’
chooseRandomMoveFromList()
possibleMoves ← []
i in movesList
self.isSpaceFree(board,i)
False
O’
True
possibleMoves.append(i)
len(possibleMoves)!=0
False
return None
True
return random.choice(possibleMoves)


O''
getComputerMove()
computerletter=='X'
False
True
playerletter ← 'X'
playerletter ← 'O'
i ← 1
J
i < 10
False
True
copy ← self.getBoardCopy(board)
N'
self.isSpaceFree(copy,i)
False
O'
True
self.makeMove(copy,computerletter,i)
L'
self.isWinner(copy, computerletter,i)
False
True
M'
return i
i ← i + 1


J
i ← 1
J'
False
i < 10
True
copy ← self.getBoardCopy(board)
N'
False
self.isSpaceFree(copy,i)
O'
True
self.makeMove(copy,playerletter,i)
L'
self.isWinner(copy, playerletter,i)
False
True
O'
return i
i ← i + 1


J'
N''
move ← self.chooseRandomMoveFromList(board,[1,3,7,9])
False
move!=None
True
return move
O'
self.isSpaceFree
(board,5)
False
True
return self.chooseRandomMoveFromList(board,[2,4,6,8])
return 5
N''


L''
isBoardFull()
i ← 1
False
return True
i < 10
True
False
self.isSpaceFree(board,i)
O'
return False
i ← i + 1

PROGRAM SOURCE CODE

```
#WORDSEARCH

from string import ascii_uppercase
from random import choice

class Wordsearch:
    right,left,down,up,downright,downleft,upright,upleft=range(8)
    def word_search_base(self,width,height):
        columns=[]
        for i in range(width):
            columns.append([choice(ascii_uppercase)for j in range(height)])
        return columns

    def insert(self,word,puzzle,x,y,direction=right, inplace=True):
        coords=[]
        if not inplace:
            new_puzzle=[[j for j in i] for i in puzzle]
        else:
            new_puzzle=puzzle
        if direction == self.right:
            i=x
            for char in word:
                if i<0:
                    raise IndexError
                new_puzzle[i][y]=char
                coords.append((i,y))
                i+=1
        elif direction == self.left:
            i = x
            for char in word:
                if i < 0:
                    raise IndexError
                new_puzzle[i][y] = char
                coords.append((i, y))
                i-= 1
```

```
elif direction == self.down:
    i = y
    for char in word:
        if i < 0:
            raise IndexError
        new_puzzle[x][i] = char
        coords.append((x, i))
        i+= 1
elif direction == self.up:
    i = y
    for char in word:
        if i < 0:
            raise IndexError
        new_puzzle[x][i] = char
        coords.append((x, i))
        i -= 1
elif direction == self.downright:
    i, j = x, y
    for char in word:
        if i < 0 or j < 0:
            raise IndexError
        new_puzzle[i][j] = char
        coords.append((i, j))
        i += 1
        j += 1
elif direction == self.downleft:
    i, j = x, y
    for char in word:
        if i < 0 or j < 0:
            raise IndexError
        new_puzzle[i][j] = char
        coords.append((i, j))
        i -= 1
        j += 1
elif direction == self.upright:
    i, j = x, y
```

```
            for char in word:
                if i < 0 or j < 0:
                    raise IndexError
                new_puzzle[i][j] = char
                coords.append((i, j))
                i += 1
                j -= 1
        elif direction == self.upleft:
            i, j = x, y
            for char in word:
                if i < 0 or j < 0:
                    raise IndexError
                new_puzzle[i][j] = char
                coords.append((i, j))
                i -= 1
                j -= 1
        return coords, new_puzzle


    def word_search(self,width, height, words):
        assert width * height > len("".join(words)), \
               "Too many words for a %sx%s puzzle" % (width, height)
        max_length = 0
        for word in words:
            if len(word) > max_length:
                max_length = len(word)
        assert max_length <= width or max_length <= height, \
                   "At least one word is too long for a %sx%s puzzle" % (width,
                    height)
        base = self.word_search_base(width, height)
        occupied = []
        # occupied coordinates
        word_coords = {}
        #Make a dictionary mapping words to their coordinates, so the
        #program knows where the words are
        # {word: (x, y)}
```

```
        for word in words:
            word = word.upper()
            while True:
                direction = choice(range(8))
                x, y = choice(range(width)), choice(range(height))
                #Insert each word at random coordinates going in a random
                #direction such that it doesn't go off the board and doesn't
                #obscure another word
                # random direction + coords
                try:
                    coords, new_puzzle = self.insert(word, base, x, y,
                                                     direction=direction,
                                                     inplace=False)
                #Keep generating random coordinates and directions and checking
                #the results (using insert with the inplace argument set
                #to False) until something works
                except IndexError:
                    # When word extends beyond the edge of the puzzle
                    continue
                else:
                    need_retry = False
                    if [c for c in coords if c in occupied]:
                        for i, j in coords:
                            if (i, j) in occupied and \
                               base[i][j] != new_puzzle[i][j]:
                                # space conflict
                                need_retry = True
                                break
                    if not need_retry:
                        self.insert(word, base, x, y, direction=direction)
                        occupied += coords
                        word_coords[word] = coords[0]
                        break
        return base, word_coords

from random import sample
```

```
from time import time


def mainprogram1():
    print "          WORD SEARCH           "
    print
    width = int(raw_input("How many columns? "))
    height = int(raw_input("How many rows? "))
    print "Loading word file..."
    print
    print "Topics"
    print "1. Animals & Birds"
    print "2. Countries"
    print "\t"
    topic=raw_input("Enter topic of your choice:")
    if topic=="Animals & Birds" or topic=='1':
        word_bank = open("animals.txt","r").read().splitlines()
    elif topic=="Countries" or topic=='2':
        word_bank = open("countries.txt","r").read().splitlines()
    word_bank = [w.lower() for w in word_bank if len(w) < width or
                 len(w) < height]
    # Filters out long words
    word_bank = sample(word_bank, int(raw_input("How many words? ")))
    print
    # Show the word bank
    print " Word Bank ".center(width * 2, "=")
    i = 0
    for word in word_bank:
        print word.center(width * 2)
    print "=" * width * 2
    w=Wordsearch()
    puzzle, word_coords = w.word_search(width, height, word_bank)
    # Display the puzzle
    for i in range(height):
        for j in range(width):
            print puzzle[j][i],
        print
```

```
start_time = time()
# Time for completing the word search
words_found = []
while len(words_found) < len(word_bank):
    print
    word = raw_input("Enter the word you found: ").lower()
    if word not in word_bank:
        print '"%s" isn\'t in the word bank.' % word
        print
        continue
    coords = raw_input("Enter the (comma-separated) coordinates
    (eg:column,row) ")
    coords = coords.strip().strip("()").strip()
    try:
        x, y = coords.split(",")
        x, y = int(x.strip()) - 1, int(y.strip()) - 1
    except ValueError, IndexError:
        print "Coordinates should be comma-separated integers."
        print
        continue
    else:
        if (x, y) == word_coords[word.upper()]:
            words_found.append(word)
            print '"%s" found. %s more to go.' % (word, len(word_bank) -
            len(words_found))
            print
        else:
            # incorrect coordinates
            print '"%s" is in the word bank, but not at those \
coordinates.' % word
            print
print '~'*77
print "Congratulations! You finished the word search in %d seconds." % (
    time() - start_time)
# Shows the time taken to complete the word search
print '~'*77
```

```
#HANGMAN

import random
HANGMANPICS = ['''

  +---+
  |   |
      |
      |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
      |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
  |   |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
 /|   |
      |
      |
=========''', '''
```

```
  +---+
  |   |
  O   |
 /|\  |
      |
      |
=========''', '''

  +---+
  |   |
  O   |
 /|\  |
 /    |
      |
=========''', '''

  +---+
  |   |
  O   |
 /|\  |
 / \  |
      |
=========''']
class Hangman:
    f1=open("animals1.txt")
    f2=open("states.txt")
    f3=open("flowers.txt")
    def getWord(self):
        print "\t"
        print "Choose from the following Topics :"
        print "1.Animals and birds"
        print "2.States of India"
        print "3.Flowers"
        print "\t"
        choice=input("Enter your choice:")
```

```
        print "\t"
        if choice ==1:
            filename="animals1.txt"
        elif choice==2:
            filename="states.txt"
        elif choice==3:
            filename="flowers.txt"
        else:
            print "Wrong Choice"
        s=random.choice(open(filename).read().splitlines())
        return s
    def Board(self,HANGMANPICS, missedLetters, correctLetters, secretWord):
        print(HANGMANPICS[len(missedLetters)])
        print
        print 'Missed letters:'
        for letter in missedLetters:
            print letter,
        print
        print "Length of the word is ",len(secretWord)
        print
        blanks = '_' * len(secretWord)
        for i in range(len(secretWord)):
            if secretWord[i] in correctLetters:
                blanks = blanks[:i] + secretWord[i] + blanks[i+1:]
        for letter in blanks:
            print letter,
        print
        print
    def getGuess(self,alreadyGuessed):
        while True:
            print 'Guess a letter.'
            guess = raw_input()
            guess = guess.lower()
            if len(guess) != 1:
                print('Please enter a single letter.')
            elif guess in alreadyGuessed:
```

```
                print('You have already guessed that letter. Choose again.')
            elif guess not in 'abcdefghijklmnopqrstuvwxyz':
                print('Please enter a LETTER.')
            else:
                return guess
    def playAgain(self):
        print('Do you want to play again? (yes or no)')
        return raw_input().lower().startswith('y')


def mainprogram2():
    print "H A N G M A N".center(48," ")
    h=Hangman()
    missedLetters = ''
    correctLetters = ''
    secretWord = h.getWord()
    gameIsDone = False
    while True:
        h.Board(HANGMANPICS, missedLetters, correctLetters, secretWord)
        guess = h.getGuess(missedLetters + correctLetters)
        if guess in secretWord:
            correctLetters = correctLetters + guess
            foundAllLetters = True
            for i in range(len(secretWord)):
                if secretWord[i] not in correctLetters:
                    foundAllLetters = False
                    break
            if foundAllLetters:
                print('Yes! The secret word is "' + secretWord + '"! You have
                won!')
                gameIsDone = True
        else:
            missedLetters = missedLetters + guess
            if len(missedLetters) == len(HANGMANPICS) - 1:
                h.Board(HANGMANPICS, missedLetters, correctLetters, secretWord)
                print('You have run out of guesses!'+', the word was "' +
                secretWord + '"')
```

```
                gameIsDone = True
        if gameIsDone:
            if h.playAgain():
                missedLetters = ''
                correctLetters = ''
                gameIsDone = False
                secretWord = h.getWord()
            else:
                break

#Tic tac toe game

import random
class tictactoe:
    def drawBoard(self,board):
        print "\n"
        print('   |   |')
        print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])
        print('   |   |')
        print('-----------')
        print('   |   |')
        print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
        print('   |   |')
        print('-----------')
        print('   |   |')
        print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
        print('   |   |')
        print "\n"

    def inputPlayerLetter(self):
        letter = ''
        while not (letter == 'X' or letter == 'O'):
            print('Do you want to be X or O?\n')
            letter = raw_input().upper()
        if letter == 'X':
            return ['X', 'O']
```

```
    else:
        return ['O', 'X']

def playsfirst(self):
    #randint returns 0 or 1, randomly choosing the first player
    if random.randint(0, 1) == 0:
        return 'computer'
    else:
        return 'player'

def playAgain(self):
    print('Do you want to play again? (yes or no)\n')
    return raw_input().lower()=='yes'

def makeMove(self,board, letter, move):
    board[move] = letter

def isWinner(self,bo, le):
    #bo=board, le=letter, returns true if the player has won
    return ((bo[1] == le and bo[2] == le and bo[3] == le) or
    (bo[4] == le and bo[5] == le and bo[6] == le) or
    (bo[7] == le and bo[8] == le and bo[9] == le) or
    (bo[7] == le and bo[4] == le and bo[1] == le) or
    (bo[8] == le and bo[5] == le and bo[2] == le) or
    (bo[9] == le and bo[6] == le and bo[3] == le) or
    (bo[7] == le and bo[5] == le and bo[3] == le) or
    (bo[9] == le and bo[5] == le and bo[1] == le))

def getBoardCopy(self,board):
    #makes a duplicate of the board list and return it the duplicate.
    dupeBoard = []
    for i in board:
        dupeBoard.append(i)
    return dupeBoard
```

```
def isSpaceFree(self,board, move):
    return board[move] == ' '


def getPlayerMove(self,board):
    move = ' '
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not
    self.isSpaceFree(board, int(move)):
        print('What is your next move? (1-9)')
        move = raw_input()
    return int(move)


def chooseRandomMoveFromList(self,board, movesList):
    possibleMoves = []
    for i in movesList:
        if self.isSpaceFree(board, i):
            possibleMoves.append(i)


    if len(possibleMoves) != 0:
        return random.choice(possibleMoves)
    else:
        return None


def getComputerMove(self,board, computerLetter):
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'


   #checking if someone won
    for i in range(1, 10):
        copy = self.getBoardCopy(board)
        if self.isSpaceFree(copy, i):
            self.makeMove(copy, computerLetter, i)
            if self.isWinner(copy, computerLetter):
                return i
```

```
        # Check if the player could win on his next move, and block them.
        for i in range(1, 10):
            copy = self.getBoardCopy(board)
            if self.isSpaceFree(copy, i):
                self.makeMove(copy, playerLetter, i)
                if self.isWinner(copy, playerLetter):
                    return i

        # Try to take one of the corners, if they are free.
        move = self.chooseRandomMoveFromList(board, [1, 3, 7, 9])
        if move != None:
            return move

        # Try to take the center, if it is free.
        if self.isSpaceFree(board, 5):
            return 5

        # Move on one of the sides.
        return self.chooseRandomMoveFromList(board, [2, 4, 6, 8])

    def isBoardFull(self,board):
        for i in range(1, 10):
            if self.isSpaceFree(board, i):
                return False
        return True

def mainprogram3():
    c=tictactoe()
    print('**** Welcome to Tic Tac Toe! ****\n\n')
    while True:
        theBoard = [' '] * 10
        playerLetter, computerLetter = c.inputPlayerLetter()
        turn = c.playsfirst()
        print('\nThe ' + turn + ' will go first.\n')
        gameIsPlaying = True
```

```
while gameIsPlaying:
    if turn == 'player':
        c.drawBoard(theBoard)
        move = c.getPlayerMove(theBoard)
        c.makeMove(theBoard, playerLetter, move)

        if c.isWinner(theBoard, playerLetter):
            c.drawBoard(theBoard)
            print('Hooray! You have won the game!\n WELL DONE!!\n')
            gameIsPlaying = False
        else:
            if c.isBoardFull(theBoard):
                c.drawBoard(theBoard)
                print('The game is a tie!\n')
                break
            else:
                turn = 'computer'

    else:
        # Computer's turn.
        move = c.getComputerMove(theBoard, computerLetter)
        c.makeMove(theBoard, computerLetter, move)

        if c.isWinner(theBoard, computerLetter):
            c.drawBoard(theBoard)
            print('The computer has beaten you! You lose.\n BETTER LUCK
            NEXT TIME!!\n')
            gameIsPlaying = False
        else:
            if c.isBoardFull(theBoard):
                c.drawBoard(theBoard)
                print('The game is a tie!\n')
                break
            else:
                turn = 'player'
```

```
        if not c.playAgain():
            break

# Main Game
while True:
    print "*"*77
    print "GAMES".center(48," ")
    print "1.Wordsearch"
    print "2.Hangman"
    print "3.Tic Tac Toe"
    print "4.Quit"
    print "\t"
    print "*"*77
    choice_no=input("Enter choice:")
    print "\t"
    print "~"*77
    if choice_no==1:
        mainprogram1()
    elif choice_no==2:
        mainprogram2()
    elif choice_no==3:
        mainprogram3()
    elif choice_no==4:
        print "Thank you for playing..... Goodbye!"
        break
    else:
        print "Wrong choice"


wait=input()
```

OUTPUT SCREEN SHOTS

```
...
********************************************************************************
                        GAMES
1.Wordsearch
2.Hangman
3.Tic Tac Toe
4.Quit

********************************************************************************
Enter choice:1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
          WORD SEARCH

How many columns? 10
How many rows? 10
Loading word file...

Topics
1. Animals & Birds
2. Countries

Enter topic of your choice:2
How many words? 5

==== Word Bank =====
       canada
        oman
        uae
       spain
       france
====================
X R J V O J S U I L
F R A N C E D L X C
S R C U D F E H A V
H E V Y S I U N Q P
G R S P I S A F B V
H X A L O D E K X U
W I I G A K U Z G K
N U I C G W F A U M
D C K N A M O G E I
E Y P B Q H N G T H
```

```
Enter the word you found: france
Enter the (comma-separated) coordinates (eg:column,row) 1,2
"france" found. 4 more to go.


Enter the word you found: spain
Enter the (comma-separated) coordinates (eg:column,row) 5,4
"spain" found. 3 more to go.


Enter the word you found: oman
Enter the (comma-separated) coordinates (eg:column,row) 7,9
"oman" found. 2 more to go.


Enter the word you found: canada
Enter the (comma-separated) coordinates (eg:column,row) 10,2
"canada" found. 1 more to go.


Enter the word you found: uae
Enter the (comma-separated) coordinates (eg:column,row) 7,4
"uae" is in the word bank, but not at those     coordinates.


Enter the word you found: uae
Enter the (comma-separated) coordinates (eg:column,row) 7,7
"uae" found. 0 more to go.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Congratulations! You finished the word search in 410 seconds.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
********************************************************************************
                    GAMES
1.Wordsearch
2.Hangman
3.Tic Tac Toe
4.Quit

********************************************************************************
Enter choice:2

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                H A N G M A N

Choose from the following Topics :
1.Animals and birds
2.States of India
3.Flowers

Enter your choice:2



  +---+
  |   |
      |
      |
      |
      |
=========

Missed letters:

Length of the word is  11

_ _ _ _ _ _ _ _ _ _ _

Guess a letter.
a


  +---+
  |   |
      |
      |
      |
      |
=========

Missed letters:

Length of the word is  11
```

```
_ _ _ a _ a _ _ _ a _

Guess a letter.
c


  +---+
  |   |
      |
      |
      |
      |
=========

Missed letters:

Length of the word is  11

_ _ _ a _ a _ c _ a _

Guess a letter.
i


  +---+
  |   |
  O   |
      |
      |
      |
=========

Missed letters:
i
Length of the word is  11

_ _ _ a _ a _ c _ a _

Guess a letter.
o


  +---+
  |   |
  O   |
  |   |
      |
      |
=========

Missed letters:
i o
Length of the word is  11

_ _ _ a _ a _ c _ a _

Guess a letter.
e


  +---+
  |   |
  O   |
 /|   |
      |
      |
=========

Missed letters:
i o e
Length of the word is  11

_ _ _ a _ a _ c _ a _

Guess a letter.
h
```

```
  +---+
  |   |
  O   |
 /|   |
      |
      |
=========

Missed letters:
i o e
Length of the word is  11

_ _ _ a _ a _ c h a _

Guess a letter.
m


  +---+
  |   |
  O   |
 /|\  |
      |
      |
=========

Missed letters:
i o e m
Length of the word is  11

_ _ _ a _ a _ c h a _

Guess a letter.
r

   +---+
   |   |
   O   |
  /|\  |
       |
       |
 =========

 Missed letters:
 i o e m
 Length of the word is  11

 _ _ _ a r a _ c h a _

 Guess a letter.
 s


   +---+
   |   |
   O   |
  /|\  |
  /    |
       |
 =========

 Missed letters:
 i o e m s
 Length of the word is  11

 _ _ _ a r a _ c h a _

 Guess a letter.
 t
```

```
  +---+
  |   |
  O   |
 /|\  |
 /    |
      |
=========

Missed letters:
i o e m s
Length of the word is  11

_ t t a r a _ c h a _

Guess a letter.
n


  +---+
  |   |
  O   |
 /|\  |
 /    |
      |
=========

Missed letters:
i o e m s
Length of the word is  11

_ t t a r a n c h a _

Guess a letter.
u


   +---+
   |   |
   O   |
  /|\  |
  /    |
       |
 =========

 Missed letters:
 i o e m s
 Length of the word is  11

 u t t a r a n c h a _

 Guess a letter.
 l
 Yes! The secret word is "uttaranchal"! You have won!
 Do you want to play again? (yes or no)
 no
```

```
********************************************************************************
                    GAMES
1.Wordsearch
2.Hangman
3.Tic Tac Toe
4.Quit

********************************************************************************
Enter choice:3

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
**** Welcome to Tic Tac Toe! ****

Do you want to be X or O?

O

The player will go first.


   |   |
   |   |
   |   |
-----------
   |   |
   |   |
   |   |
-----------
   |   |
   |   |
   |   |


What is your next move? (1-9)
1


   |   |
 O |   |
   |   |
-----------
   |   |
   |   |
   |   |
-----------
   |   |
   |   | X
   |   |


What is your next move? (1-9)
5
```

```
   |   |
 O |   | X
   |   |
-----------
   |   |
   | O |
   |   |
-----------
   |   |
   |   | X
   |   |


What is your next move? (1-9)
6


   |   |
 O |   | X
   |   |
-----------
   |   |
 X | O | O
   |   |
-----------
   |   |
   |   | X
   |   |
What is your next move? (1-9)
2


   |   |
 O | O | X
   |   |
-----------
   |   |
 X | O | O
   |   |
-----------
   |   |
   | X | X
   |   |


What is your next move? (1-9)
7


   |   |
 O | O | X
   |   |
-----------
   |   |
 X | O | O
   |   |
-----------
   |   |
 O | X | X
   |   |
```

```
The game is a tie!

Do you want to play again? (yes or no)

YES
```

```
Do you want to be X or O?

X

The player will go first.



   |   |
   |   |
   |   |
-----------
   |   |
   |   |
   |   |
-----------
   |   |
   |   |
   |   |


What is your next move? (1-9)
7


   |   |
   |   |
   |   |
-----------
   |   |
   |   |
   |   |
-----------
   |   |
 X |   | O
   |   |

What is your next move? (1-9)
6


   |   |
 O |   |
   |   |
-----------
   |   |
   |   | X
   |   |
-----------
   |   |
 X |   | O
   |   |
```

```
What is your next move? (1-9)
5


   |   |
 O |   | O
   |   |
-----------
   |   |
   | X | X
   |   |
-----------
   |   |
 X |   | O
   |   |


What is your next move? (1-9)
4


   |   |
 O |   | O
   |   |
-----------
   |   |
 X | X | X
   |   |
-----------
   |   |
 X |   | O
   |   |


Hooray! You have won the game!
 WELL DONE!!

Do you want to play again? (yes or no)

no
*********************************************************************************
                      GAMES
1.Wordsearch
2.Hangman
3.Tic Tac Toe
4.Quit

*********************************************************************************
Enter choice:4

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Thank you for playing..... Goodbye!
```

BIBLIOGRAPHY

https://inventwithpython.com/
http://www.pygame.org/wiki/tutorials
https://www.python.org/