U of A

CMPUT174 Final EXAM STUDY GUIDE

Lecture Notes

```
# Remember The Word Version 4 - USER DEFINED FUNCTIONS

# This program does the following:

# create the window, display the icon and display instructions

# prompt player to press enter

# display words

# get guess

# display results - output win or condolence message but not both

# end game by closing the window

# replace adjacent duplicate line groups  with a repetition

# control structure - for statement

# limit the occurrence of literals

# Main Algorithm

import time

from uagame import Window

def main():



    # display icon

    display_icon(window) # function call

    # display instructions

    text_size = 24

    text_color = 'white'

    display_instructions(window,text_size,text_color) # function call


    # CLEAR WINDOW
```

```
window.clear()

# DISPLAY ICON

display_icon(window) # function call


# display words


# create word list

word_list = ['orange','chair','mouse','sandwich']

correct_answer = word_list[1]

start_letter = correct_answer[0]

# FUNCTION CALL to display the word

display_words(window,text_size,text_color,word_list)




# get guess - FUNCTION CALL

guess = get_guess(window,text_size,text_color,start_letter)


# CLEAR WINDOW

window.clear()

# DISPLAY ICON

display_icon(window) # function call

# display results - FUNCTION CALL

display_results(window,text_size,text_color,guess,correct_answer)


de
```

```
# USER DEFINED FUNCTIONS

def end_game(window):

    font_height = window.get_font_height()

    x = 0

    window_height = window.get_height()

    y = window_height - font_height

    prompt2 = 'Press enter to end the game.'

    window.input_string(prompt2,x,y)

    window.close()

def display_results(window,text_size,text_color,guess,correct_answer):

    x = 0

    y = 0

    window.set_font_size(text_size)

    window.set_font_color(text_color)

    font_height = window.get_font_height()

    win_message = 'Congratulations, you are correct.'

    lose_message = 'Sorry you entered ' + guess

    correct_answer_message = 'The correct answer was '+ correct_answer

    if guess == correct_answer:

        window.draw_string(win_message,x,y)

    else:

        window.draw_string(lose_message,x,y)

    y = y + font_height

    window.draw_string(correct_answer_message,x,y)
```

```
def get_guess(window,text_size,text_color,start_letter):
    x = 0
    y = 0
    window.set_font_size(text_size)
    window.set_font_color(text_color)
    prompt_for_word = 'What word starts with the letter '+start_letter+'?'
    guess = window.input_string(prompt_for_word,x,y)
    return guess
def display_words(window,text_size,text_color,word_list):
    x = 0
    y = 0
    pause_time = 2
    for word in word_list:
        window.set_font_size(text_size)
        window.set_font_color(text_color)
        window.draw_string(word,x,y)
        time.sleep(pause_time)
        # CLEAR WINDOW
        window.clear()
        # DISPLAY ICON
        display_icon(window) # function call
def display_icon(window):
    # display an icon
    # - window is uagame.Window object on which the icon is displayed
```

```
        icon_color = 'green'

        icon_size = 100

        icon_string = 'UA'

        window.set_font_color(icon_color)

        window.set_font_size(icon_size)

        window_width = window.get_width()

        icon_width = window.get_string_width(icon_string)

        x_icon = window_width - icon_width

        y_icon = 0

        window.draw_string(icon_string,x_icon,y_icon)


def display_instructions(window,text_size,text_color):

        # displays the instructions

        # -window is uagame.Window object on which the instruction are displayed

        # -text_size int object for text size

        # -text_color is str object for text color

        window.set_font_size(text_size)

        window.set_font_color(text_color)

        x = 0

        y = 0

        font_height = window.get_font_height()

        # create instruction list

        instruction_list = ['A sequence of words will be displayed','You will be asked which words starts 
with','a particular letter.','You win if you enter the right word.']


        # Use for statement to display the instructions
        for instruction in instruction_list:

            window.draw_string(instruction,x,y)

            y = y + font_height


        # prompt player to press enter

        prompt1 = 'Press the enter key to display the words.'

        window.input_string(prompt1,x,y)


    main()
```

CMPUT175-Lab 3 Exercises

The solutions to these exercises have to be shown to your TA in your assigned Lab before the end of your lab time.

The exercises are about building the game of Tic Tac Toe. Tic Tac Toe is a game for two players who take turns to fill a 3 X 3 grid with either o or x. Each player alternates choosing an open space in the grid to mark with either x for a player or o for the other player. The goal (to win) is to have three xs or 3 os in a row, column or diagonal. The game can also end with a draw if the grid is full but no line is formed. The exercises combined will allow us to implementing Tic Tac Toe for two human players to play it on a computer.

1. Write a program that displays the Tic Tac Toe board. We have a python class in the provided file **Lab3Exercise.py** that initializes a board with a list of empty characters.

```
class TicTacToe:
    def __init__(self):
        # "board" is a list of 10 strings representing the board (ignore index 0)
        self.board = [" "]*10
        self.board[0]="#"
```

The constructor **`__init__`** creates a list of 10 one character strings. The first one at index 0 will be ignored (initialized with some character but not space) and the other 9 represent the 9 cells of the Tic Tac Toe grid. Write the method **`assignMove(self, cell,ch)`** that requests the Tic Tac Toe object to assign the symbol/letter **ch** to the position cell of the grid. The argument **ch** is assumed to be either “x” or “o”. However, **cell** is a number and should be between 1 and 9. Write the method **`drawBoard(self)`** that would display the board as follows:

```
 x | x | o        7 | 8 | 9
-----------      -----------
 o | o | x        4 | 5 | 6
-----------      -----------
 x | o | x        1 | 2 | 3
```

The first grid on the left represents the current state of the board while the grid on the right is static, and represents the index of each cell (like the numeric pad on the keyboard). The two grids are separated by a tab (\t).

Once you wrote your methods, create an object Tic Tac Toe with **`myBoard=TicTacToe()`** fill some cells in the board using for example **`myBoard.assignMove(4,”x”)`** to put an x in the 4th cell, then display the board.

If the list board of the instance TicTacToe contains [“#”, “x”,”o”,”x”,”o”,”o”,”x”,”x”,”x”,”o”], the board should look as above. If the list contains [“#”, “o”,”o”,”o”,”x”,”x”,”x”,”x”,”x”,”o”] the board should look like:

```
 x | x | o        7 | 8 | 9
-----------      -----------
 x | x | x        4 | 5 | 6
-----------      -----------
 o | o | o        1 | 2 | 3
```

If the list contains [“#”, “o”,”o”,”o”,”o”,” ”,”o”,”o”,”o”,”o”] the board should look like:

```
 o | o | o        7 | 8 | 9
-----------      -----------
 o |   | o        4 | 5 | 6
-----------      -----------
 o | o | o        1 | 2 | 3
```

Write the methods **`boardFull(self)`** to check if the board is full, and **`cellIsEmpty(self, cell)`** to check whether a given cell is available. Make sure the argument cell is correct and will not generate an error of out of bound. If it is out of bound, the method should return **False**.

Write a small program to use and test these methods.

2. Continue the class TicTacToe by completing the method **`isWinner(self, ch)`** which returns True if the symbol in **ch** forms either a horizontal line, a vertical line or a diagonal line, and False otherwise. A line is formed by three consecutive cells with the same symbol (as in **ch**). This method is called within the method **`whoWon(self)`** provided in the python file.

Write a small program that uses and test these methods to fill the cells with **`assignMove`** check whether the board is full, whether a given cell is available, who won and display the board.

3. Write a program that allows two players, x and o, to alternate and provide their moves on the board. For instance asking the question: **It is the turn for x . What is your move?**
The program should check the input and only allow a valid cell between 1 and 9. Also, a cell that is not available (already filled) should not be allowed as a new move. The program should iterate until the game is over and display who won or whether there is a draw.
Here are some examples of output at the end of a game.

```
 x | x | o         7 | 8 | 9
-----------       -----------
 o | o | x         4 | 5 | 6
-----------       -----------
 x | o | x         1 | 2 | 3
It's a tie.

 x | o | o        7 | 8 | 9
-----------       -----------
 x | x | o        4 | 5 | 6
-----------       -----------
   |   | x        1 | 2 | 3
x wins. Congrats!
```

CMPUT175-Lab 2 Exercises

The solutions to these exercises have to be shown to your TA in your assigned Lab before the end of your lab time.

To get full marks for this lab, you must complete Exercise 1 and either exercise 2 or 3 (choose one).

1. Write a program that reads in a list of accounts from a file, and then allows the user to enter transactions for each account. The file **accounts.txt** contains a list of accounts with initial balances. As the program runs, it should ask the user for the name of an account, and the amount of money to deposit or withdraw from the account. After each transaction, the program should print the amount remaining in the account. If the **accounts.txt** file is missing, the program should exit with an error message. If any account has a non-float value for the balance, the program should display an error and not add that account. If a non-float is entered for a transaction, or if the account name entered does not exist, the program should print an error message and cancel the transaction. The program should exit when the user enters "**Stop**" for an account name.

Note that the **accounts.txt** file DOES NOT need to be updated by the program.

Input File Example

```
Bob:234.70

Tom Johnson:791.56

Anna:3260.55

Jill:-23.76

Mike Smith:Not_A_Float!

. . .
```

When the line:

Mike Smith:Not_A_Float!

is encountered, the program should print: **Account for Mike Smith not added: illegal value for balance**

If the input file is missing, the program should print: **Input file not found, program will exit**

An example of interaction with the program could look like this:

```
Enter account name, or 'Stop' to exit: Bob

Enter transaction amount: -45.23

New balance for account Bob: 189.47

Enter account name, or 'Stop' to exit: Bob

Enter transaction amount: 117.90
```

```
New balance for account Bob: 307.37

Enter account name, or 'Stop' to exit: Anna

Enter transaction amount: this_is_not_a_float

Illegal value for transaction, transaction canceled

Enter account name, or 'Stop' to exit: Mike Smith

Account does not exist, transaction canceled

Enter account name, or 'Stop' to exit: Stop
```

2 . Write a program that completes the incomplete python class Automobile in the file **Lab2Exercise2.py**, by implementing the methods **get_horsepower()**, **get_color()**, and **get_worth()**. This program should run the **main()** function, which will ask the user for the length, horsepower, and color of an automobile, and will thenprint out the worth of that automobile. An automobile's worth in dollars is calculated as follows:

worth = horsepower*length*color_factor*10

Where color_factor is 3 if the automobile's color is “red”, 2 if the automobile's color is “yellow” or “blue”, and 1 otherwise. For this program, you must use the Automobile class and call the **get_worth()** method. This program does not need to do any exception handling.

```
Example interaction with the program:

Enter automobile's length in meters: 3

Enter automobile's horsepower: 200

Enter automobile's color: white

This automobile is worth $6000.
```

3. Write a number guessing game. The program will start by choosing a random number between 1 and 20 (inclusive). The user has 6 chances to guess what the number is. The program should end when either the user guesses the correct number, or else they have no guesses left. A message should inform the user if they guessed correctly or ran out of guesses. When the user enters a guess, the program must check that the guess is between 1 and 20 (inclusive).

If the guess is not, the program should print a warning message and ask the user to guess again (this should not count as a guess). After each guess, the program should tell the user if the guess is too high or too low. This program does not need to do any exception handling.

Optional – not for marks: Ask the user if they want to play again when they have either made a correct guess or run out of guesses, and then generate a new number and run the game again.

Example interactions with program could look like:

```
Enter a guess (1-20): 10

Too low!

Enter a guess (1-20): 15

Too high!

Enter a guess (1-20): 13

Correct! The number was 13.


Enter a guess (1-20): 14

Too high!

Enter a guess (1-20): 45

That number is not between 1 and 20!

Enter a guess (1-20): 9

Too high!

Enter a guess (1-20): 1

Too low!

Enter a guess (1-20): 4

Too low!

Enter a guess (1-20): 7

Too high!

Enter a guess (1-20): 6

Too high!

You are out of guesses. The number was 5.
```

Lab 4 Exercises

Problem 1 - Web Browser Navigation

Every web browser has both a back and a forward button which allows the user to navigate to previously seen web pages.

Your task is to implement this functionality using two stacks (i.e. a simulation):

The interaction with your program will be a sequence of commands, either ">", "<", or "=". The ">" and "<" tokens will be used to indicate forward and back, respectively. The "=" is to indicate a new current page. After the "=", the computer would expect a web address. You can assume the web addresses are always valid.

You will not actually have to do any web navigation and you will simply be keeping track of the current page by outputting the given address between "[" and "]".

There are a few situations you should take care to handle.

First, if a user enters "<" or ">" without there being a previous or next page to go to, you should output a corresponding error message but allow the user to continue browsing.

Secondly, if a user enters "<" and then a new web address, the user's previous browsing history in the ">" direction should be erased.

For example, if a user is on "www.google.com", enters "<", and then enters "www.yahoo.com", the history of visiting "www.google.com" should be lost. This means that if the user then enters ">", this should generate an error message and not "www.google.com". This behaviour can also be seen below in the sample data. You can also test an actual web browser (specifically Firefox or Chrome) if you feel there is other ambiguity. Also, keep in mind that navigation should start with a home page which we will set as "www.cs.ualberta.ca".

The interaction should start with the current page being www.cs.ualberta.ca

You may choose how your program receives the input.

Problem 2 - Solving a Maze

Write a program that uses the Stack implementation given to you in stack.py to check if there is a solution to a maze. A simple maze implementation is given to you in maze.py. To solve a maze using the Stack, use the following algorithm:

Add the start square to the stack.

Repeat the following as long as the stack is not empty:

Pop a square off the stack (the current square)

If the current square is the finish square, the solution has been found

Otherwise, get the list of squares which can be moved to from the current square, and add them to the stack

Maze Description:

The maze.py file has two classes: Maze and MazeSquare.

You can create a new maze by calling the constructor: Maze(file_name), where file_name is the name of a file containing a maze.

You can get the start square of the maze by calling maze.get_start_square(). This method returns a MazeSquare.

To check if a square is the finish square, call maze.is_finish_square(square), where square is a MazeSquare.

The MazeSquare class represents a single square in the Maze. Calling maze_square.get_legal_moves() returns a list of MazeSquares that can be moved to.

Type or Class

Built-in	**Imported**	**User Defined - these are some examples of what we made in class**
int	uagame.Window	Game
float	pygame.Surface	Dot
bool	pygame.Rect	
str	pygame.Color	
list		
tuple		
range		
NoneType		
module		
function		
method		

Statement Kinds

expression
assignment
function/method definition
import
if
while
for
return
class definition

Token Kinds

keyword
identifier
literal (there are str literals, int literals, float literals)
operator
delimiter

Friends

Title on YouTube: Friends - The Secret Closet [Joey & Chandler]

URL: http://www.youtube.com/watch?v=R4VCrlWyXbQ (0:00-1:12)

Chandler:	I was trying to open your closet! I wasn't trying to open your closet, I swear!
Joey:	Wow. Monica runs a pretty tight ship around here, doesn't she? What are you doing?
Chandler:	Monica has a secret closet and she won't let me see what's in it.
Joey:	Why not?
Chandler:	I don't know! What could she possibly be hiding in here that I can't see?
Joey:	I don't know….ooohhhh I bet it's Richard.
Chandler:	What?
Joey:	Before you say I'm wrong, let me ask you something. When was the last time you saw him?
Chandler:	Why would Monica be keeping Richard in here?
Joey:	Well, off the top of my head, uh, maybe she's having her cake and eating it too. You being the cake, Richard being the too.
Chandler:	That was off the top of your head?
Joey:	Yeah. Or…
Chandler:	And here we go…
Joey:	I saw this movie once, where there was a door and nobody knew what was behind it. And when they finally got it open, millions and millions and millions of bugs came pouring out and they feasted on human flesh. You know it wouldn't kill you to respect your wife's privacy!

What is BPA, and what are the concerns about BPA?

Answers from Brent A. Bauer, M.D.

BPA stands for bisphenol A. BPA is an industrial chemical that has been used to make certain plastics and resins since the 1960s.

BPA is found in polycarbonate plastics and epoxy resins. Polycarbonate plastics are often used in containers that store food and beverages, such as water bottles. They may also be used in other consumer goods.

Epoxy resins are used to coat the inside of metal products, such as food cans, bottle tops and water supply lines. Some dental sealants and composites also may contain BPA.

Some research has shown that BPA can seep into food or beverages from containers that are made with BPA. Exposure to BPA is a concern because of possible health effects of BPA on the brain, behavior and prostate gland of fetuses, infants and children. Additional research suggests a possible link between BPA and increased blood pressure.

However, the Food and Drug Administration (FDA) has said that BPA is safe at the very low levels that occur in some foods. This assessment is based on review of hundreds of studies.

The FDA is continuing its review of BPA, including supporting ongoing research. In the meantime, if you're concerned about BPA, you can take these steps to reduce your exposure:

Use BPA-free products. Manufacturers are creating more and more BPA-free products. Look for products labeled as BPA-free. If a product isn't labeled, keep in mind that some, but not all, plastics marked with recycle codes 3 or 7 may be made with BPA.

Cut back on cans. Reduce your use of canned foods since most cans are lined with BPA-containing resin.

Avoid heat. The National Institute of Environmental Health Sciences, part of the National Institutes of Health, advises against microwaving polycarbonate plastics or putting them in the dishwasher, because the plastic may break down over time and allow BPA to leach into foods.

Use alternatives. Use glass, porcelain or stainless steel containers for hot foods and liquids instead of plastic containers.

Why We Need to Save Our Oceans Now—Not Later

by Jose Vicente Troya (United Nations), 2017

UNITED NATIONS, Jun 01 (IPS) - What if the blue fades way as seawaters become brown andcoral reefs become white as marine grasslands wither and life below water vanishes?

This is already happening at a staggering rate. It's a lose-lose for all: people and planet.

Fish stocks are declining. Around 80 percent of fishing is either collapsing or just fully exploited. The ocean is also being polluted at an alarming rate. Fertilizer run-off and 10 to 20 million metric tons of plastic debris enter the oceans each year and destroy biodiversity and ecosystems.

At this rate the number of dead zones will increase and by the year 2050 the oceans could contain more plastic than fish, measured by weight.

If we don't take action now this trend may become irreversible. Recognizing this urgency, country representatives will gather at the Ocean Conference, 5 to 9 June at the UN headquarters in New York to address marine pollution, declining fisheries, loss of coastal and marine habitat and the vanishing life below water.

The Conference will focus will be on the Sustainable Development Goal (SDG) 14, to conserve and sustainably usethe oceans, seas and marine resources. This SDG along with 16others compose the sustainable development agenda globally adopted in 2015.

While several of the goals are to be achieved by the year 2030, most of the ocean-related targets must be attained by 2020 if we are to save our Government commitments are crucial now. They range from sustainably managing marine and coastal ecosystems to effectively regulating harvesting and end overfishing or unregulated fishing.

Governments also need to conserve at least 10 percent of coastal and marine areas; and, prohibit certain forms of fisheries subsidies, which contribute to overcapacity and overfishing.

We need to act now.

Several innovative and inspiring practices are taking place in the world which could be shared and scaled-up. Latin American and Caribbean countries are cooperating to administrate multi-country marine ecosystems, which require a shared management.

With the support of the United Nations Development Programme (UNDP) and the Global Environment Facility (GEF), several countries in the region sharing large marine ecosystems (LMEs) such as the Caribbean Sea and the Humboldt Current System are jointly working towards addressing key aspects of the market forces that drive overfishing and weak governance leading to fisheries overexploitation and degradation of coastal and marine biodiversity.

They are also seeking to ensure the conservation and sustainable delivery of LMEs goods and services, which are essential to the livelihoods of local communities, national economies and life below water.

These efforts are taking place in a region where marine ecosystems are the pillar for domestic economies, particularly in the case of Small Island Development States (SIDS). Latin America and the Caribbean have 746 marine protected areas covering 300,000 km2 and several countries have committed to expanding them.

Caribbean countries, recognizing the key role that the seas play for their future, are mainstreaming oceans into their national development planning and now are talking on adopting the blue economy paradigm.

This is a development framework to foster equity in the access to, development of and sharing of benefits from marine resources, ensuring their conservation and sustainable use, as well as reinvesting in human development. The Caribbean SIDS have pledged to protect 20 percent of their coastal and marine zones.

In preparation to the Ocean Conference, UNDP is supporting 25 national consultations to identify and register contributions made by national governments, civil society and private sector towards the conservation and sustainable use of oceans (SDG 14).

Nine of these consultations are being held in the Latin America and Caribbean region and national stakeholders are sharing their best practices on pollution reduction, support to sustainable fisheries and coastal communities, conservation and sustainable use of coastal and marine biodiversity, and removal of subsidies harming seas and their resources.

Now is the time to save our ocean and all lives that depend on it. We need to act now, before the blue fades away.