BINUS UNIVERSITY
BINUS INTERNATIONAL

Assignment Cover Letter

(Individual Work)

Student Information:	**Surname**	**Given Names**	**Student ID Number**
	Galastu	**Chandra Utama**	**2101710920**

Course Code	**: COMP6502**	**Course Name**	**: Introduction to Programming**
Class	**: L1AC**	**Name of Lecturer(s)**	**:** 1. Ida Bagus Kerthyayana 2. Tri Asih Budiono
Major	**: CS**		

Title of Assignment: **Snake**

Type of Assignment: **Final Project**

Submission Pattern

Due Date	**: 6-11-2017**	**Submission Date**	**: 6-11-2017**

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

1. Galastu Chandra Utama H

Snake Game

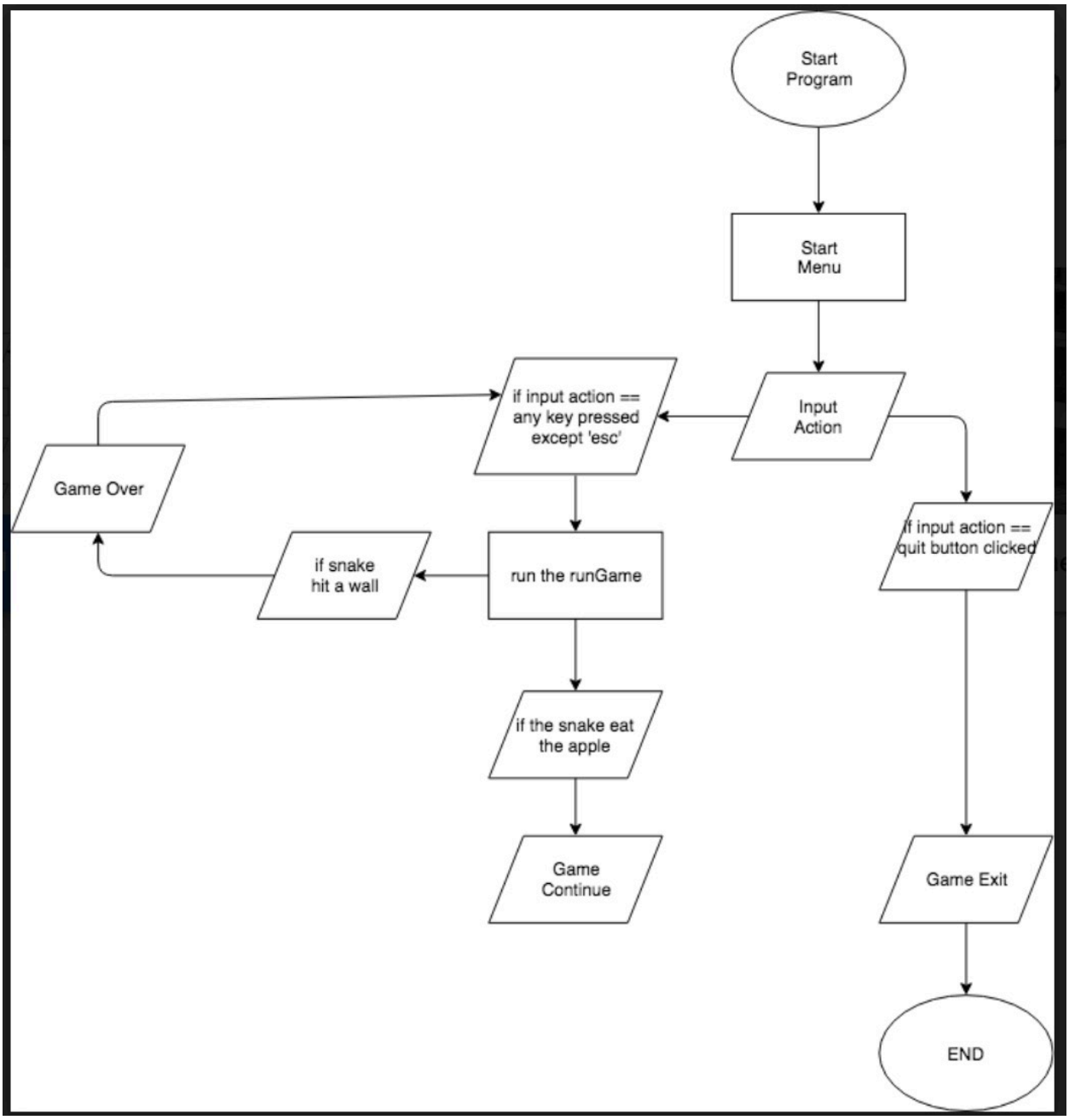
Name : Galastu Chandra Utama H

ID : 2101710920

I. Description

This program is developed by using pygame, this game is about a classic game called snake. In the making of this game I want to try to bring back the old classic game. How to play this game is simply just using arrow keys or WASD keys. There is a score counter and if you hit a wall you will lose the game.

II. Flowchart



III. Explaination of each function

- **The main() Function**

```
def main():
    global FPSCLOCK, DISPLAYSURF, BASICFONT

    pygame.init()
    FPSCLOCK = pygame.time.Clock()
    DISPLAYSURF = pygame.display.set_mode((WINDOWWIDTH, WINDOWHEIGHT))
    BASICFONT = pygame.font.Font('geektastic.ttf', 18)
    pygame.display.set_caption('SNAKEZ')

    showStartScreen()
    while True:
        runGame()
        showGameOverScreen()
```

In the Snake game program, I've make the main part of thecode in a function as runGame(). Because I want to show the "start screen" before the game start. Once when the program starts theni want to call runGame(), that will start the Snake game.

- **Separate runGame() Function**

```
-   def runGame():

        pygame.mixer.music.load("Snake_Theme_song.wav")
        pygame.mixer.music.play(-1)

        startx = random.randint(5, CELLWIDTH - 6)
        starty = random.randint(5, CELLHEIGHT - 6)
        wormCoords = [{'x': startx,     'y': starty},
                      {'x': startx - 1, 'y': starty},
                      {'x': startx - 2, 'y': starty}]
        direction = RIGHT


        apple = getRandomLocation()
```

in the beginning of the game, we want to place the starting snake at a random place but not to close to the edge so it wont get auto lose streak. So we store a random coordinate in startx and starty.

There will be one dictionary value per body segment of the worm. The dictionary will have keys 'x' and 'y' for the XY coordinates of that body segment.

- The event handling loop

```
-   while True:
        for event in pygame.event.get():
            if event.type == QUIT:
                terminate()
            elif event.type == KEYDOWN:
                if (event.key == K_LEFT or event.key == K_a) and
    direction != RIGHT:
                    direction = LEFT
                elif (event.key == K_RIGHT or event.key == K_d) and
    direction != LEFT:
                    direction = RIGHT
                elif (event.key == K_UP or event.key == K_w) and
    direction != DOWN:
                    direction = UP
                elif (event.key == K_DOWN or event.key == K_s) and
    direction != UP:
                    direction = DOWN
                elif event.key == K_ESCAPE:
                    terminate()
```

This is the start of the game loop and the start of the event handling loop. If the event is a QUIT event, then call it terminate()

If the event is a KEYUP event, then we check if the key that was pressed down is an arrow or a WASD key. In this case we want an additional check so that the worm does not turn in on itself. To avoid the worm to accidientally hit it self when the player press the opposite direction.

- Collision Detection

```
-   if wormCoords[HEAD]['x'] == -1 or wormCoords[HEAD]['x'] == CELLWIDTH
    or wormCoords[HEAD]['y'] == -1 or wormCoords[HEAD]['y'] ==
    CELLHEIGHT:
        return
    for wormBody in wormCoords[1:]:
        if wormBody['x'] == wormCoords[HEAD]['x'] and wormBody['y'] ==
    wormCoords[HEAD]['y']:
            return
```

The worm will crash when the head already moved over the edge of the grid or when the head hit another body segment.

If the head has moved off the edge of the grid by seeing if either the X coordinate of the head is -1 or equal to CELLWIDTH

The head has also moved off the grid if the Y coordinate of the head is either -1 or CELLHEIGHT

If this happen the “Game Over” screen will show.

- Detecting Collisions with the Apple

```
-   if wormCoords[HEAD]['x'] == apple['x'] and wormCoords[HEAD]['y'] ==
    apple['y']:

        apple = getRandomLocation()
    else:
        del wormCoords[-1]
```

This is a similar collision detection check between the head of the worm and the apple's XY coordinates. If they match, we set the coordinates of the apple to a random new location.

If the head has not collided with an apple, then we delete the last part of the body segment in the wormCoords list.

This part of a code will add a new body segment in the direction that the worm is going to. This will make the worm one pixel longer.

- Moving the worm

```
-   if direction == UP:
        newHead = {'x': wormCoords[HEAD]['x'], 'y': wormCoords[HEAD]['y']
    - 1}
    elif direction == DOWN:
        newHead = {'x': wormCoords[HEAD]['x'], 'y': wormCoords[HEAD]['y']
    + 1}
    elif direction == LEFT:
        newHead = {'x': wormCoords[HEAD]['x'] - 1, 'y':
    wormCoords[HEAD]['y']}
    elif direction == RIGHT:
        newHead = {'x': wormCoords[HEAD]['x'] + 1, 'y':
    wormCoords[HEAD]['y']}
    wormCoords.insert(0, newHead)
```

To move the worm, I add a new body segment to the beginning of the wormCoords list. Because the body segment is being added to the beginning of the list, it will become the new head. The coordinates of the new head will be right nex to the old head's coordinates.

- Drawing the screen

```
    DISPLAYSURF.fill(BGCOLOR)
    drawGrid()
    drawWorm(wormCoords)
    drawApple(apple)
    drawScore(len(wormCoords) - 3)
    pygame.display.update()
    FPSCLOCK.tick(FPS
```

The code for drawin the screen in the runGame() function is quite simple. In the entire display Surface with the background color. Draw the grid, worm, apple, and score to the display Surface. Then the call to pygame.display.update() draws the display Surface to the actual screen.

- Drawin “Press a key”

```
    def drawPressKeyMsg():
        pressKeySurf = BASICFONT.render('Press a key to play.', True,
    WHITE)
        pressKeyRect = pressKeySurf.get_rect()
        pressKeyRect.topleft = (WINDOWWIDTH - 200, WINDOWHEIGHT - 30)
        DISPLAYSURF.blit(pressKeySurf, pressKeyRect)
```

When the screen animation is playing or the game over screen is being shown, there will be some small text in the bottom right corner that says “Press a key to play.”

- The checkForKeyPress() Function

```
    def checkForKeyPress():
        if len(pygame.event.get(QUIT)) > 0:
            terminate()

        keyUpEvents = pygame.event.get(KEYUP)
        if len(keyUpEvents) == 0:
            return None
        if keyUpEvents[0].key == K_ESCAPE:
            terminate()
        return keyUpEvents[0].key
```

This function first checks if there are any QUIT events in the event queue. The call to pygame.event.get() returns a list of all the QUIT events in the event queue. If there are not QUIT events in the event queue, then the list that pygame.event.get() returns will be the empty list:[]