<u>Pseudocode</u>
<u>Connor Hutchinson</u>
<u>23029981</u>

```
BEGIN
Import sys
Create class sides with variables side1 and side2
        side1 contains an array with “Student”, “Dog”, “Cat”, “Hamster”.
        side2 contains an array with no data

Define main function
Print “A student finds herself in a riverbank with three pets: a dog, a cat, and a hamster. She needs to transport all three to the other side of the river in her boat. However, the boat has room for only the student herself and one pet (either the dog, the cat, or the hamster). In her absence, the dog would attack the cat, and the cat would attack the hamster."
Create boolean value win set to false
While win is false
If checkForConflicts != None then
Break
Win = checkForWin
If win == false then
Print "Currently, side 1 has " + str(sides.side1) + ". Side2 has " + str(sides.side2”
CheckTargetAndLocation
        If win == True then
print “You win”
sys.exit

Define moveAnimal with variables target and location
If target == “Student” then
if location == “side1” then
Remove target from side1
Append target onto side2
else
                        Remove target from side2
Append target onto side1
else
if location == side1 then
if ”Student” not in sides.side1 then
print “Student is not on side 1 so they cannot move the animal ”
else
                                Remove “Student” from sides.side1
                                Append “Student” onto sides.side2
Remove target from side1
Append target onto side2
                        else
                                if "Student" not in sides.side2:
                                Print “Student is not on side 2 so they cannot move "
                                else
                                Remove target from sides.side2
                                Append target onto sides.side1
                                Remove student from sides.side2
                                Append student onto sides.side1

Define checkForWin
        if "Student" in sides.side2 and "Dog" in sides.side2 and "Cat" in sides.side2 and "Hamster" in sides.side2 then
```

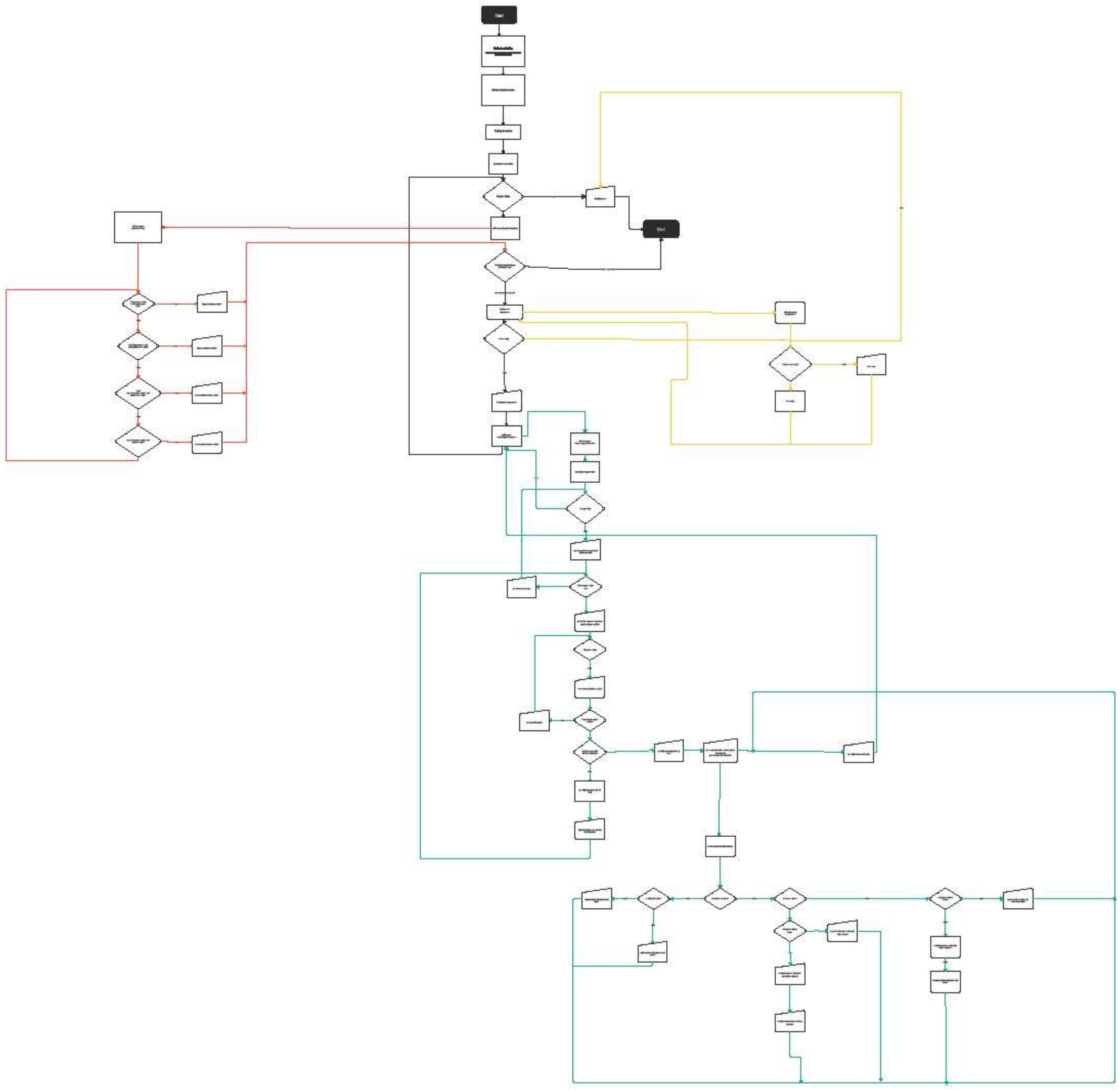
```
                Win = true
        Else
        Win = false
        Return win

Define checkForConflicts
        if "Dog" in sides.side1 and "Cat" in sides.side1 and "Student" not in sides.side1 then
                print "Dog attacked Cat on side1"
                sys.exit
        Else if "Dog" in sides.side2 and "Cat" in sides.side2 and "Student" not in sides.side2 then
                print "Dog attacked Cat on side2"
                sys.exit
        Else if "Cat" in sides.side1 and "Hamster" in sides.side1 and "Student" not in sides.side1 then
                print "Cat attacked Hamster on side1"
                sys.exit
        Else if "Cat" in sides.side2 and "Hamster" in sides.side2 and "Student" not in sides.side2 then
                print "Cat attacked Hamster on side2"
                sys.exit

Define checkTargetAndLocation
Create boolean target = false
While target == false then
userTarget=input “What target would you like to move?”
        if userTarget in sides.side1 or userTarget in sides.side2 then
target = true
                Location = false
        While location == false then
        userLocation ==”side1” or userLocation ==”side2” then
        if UserTarget not in sides and userLocation then
        location = false
        print “That target is not on that side”
else
location = true
moveAnimal userTarget to userLocation
win = false
else
print “That is not a side”
else
print ”Sorry that is not a target”

END
```

Flowchart
Haris Kadu
22009453
Please note that we can send this separately if required, as the PDF for this has already been generated.



Complexity

Hitesh Lad
23011542

```
#create class called sides which has attributes of both sides of the river (comment on the code)
class sides:
    side1 = ["Student","Dog","Cat","Hamster"] 0(1)
    side2 = []  0(1)
```

The time complexity is constant as it involves assigning values to the two variables. The space complexity is 0(1)) as it is defining two fixed sides lists. 0(n)

```
#the main function which will run 0(1)
def main(): 0(1)
```

0(1) constant complexity

```
    #display question to user (comment on the code)
    print("A student finds herself in a riverbank with three pets: a dog, a cat, and a hamster. She needs to transport all three to the other side of the river in her boat. However, the boat has room for only the student herself and one pet (either the dog, the cat, or the hamster). In her absence, the dog would attack the cat, and the cat would attack the hamster.")  0(1)  constant complexity
    #set boolean to false to allow while loop to run (comment on the code)
    win = False 0(1)
```

The time complexity is constant. The space complexity is also constant at 0(1)

The time is constant for both the function and the variable. The space complexity is 0(1))

```
    #while loop which runs while win is false. (comment on the code)
    while win == False: 0(1)
```

The time complexity is dependent on the users input and if the solution the player chose is correct or incorrect. The space is constantly 0(1)

```
        #if checkForConflicts function returns anything then the code ends (comment on the code)
        if checkForConflicts() != None: 0(1)
            break
```

The time complexity in this case calls upon the checkForConflicts function, which is defined further down. This is constant. The space is constant at 0(1)

```
        #function to check if the user has won (comment on the code)
        win = checkForWin() 0(1)
        if win == False: 0(1)
```

Constant complexity 0(1)

```
            #display what each side contains (comment on the code)
            print("Currently, side 1 has " + str(sides.side1) + ". Side2 has " + str(sides.side2)) 0(n)
            #call function to check if target picked and side picked are valid (comment on the code)
```

0(1) constant complexity as it is a 1 step or constant

```
            checkTargetAndLocation() 0(n)
        #if win is true the code ends (comment on the code)
        if win == True: 0(1)
            input("You win")  0(1)
```

0(1) constant complexity as it is a 1 step or constant

```
                def moveAnimal(target, location):  0(1)
        #if the user has chose to move only student then it will go down this path as it does not contain the additional append and remove of student  (comment on the code)
```

Constant complexity 0(1)

```
        if target == "Student": 0(n)
            #this if statement is to make sure the animal is moved from the right side (comment on the code)
```

The time is constant as it checks conditions and prints messages. The space is also constant at 0(1)

```
    if location == "side1": 0(n)
        #moves the animal from side 1 (comment on the code)
        sides.side1.remove(target) 0(n)
        sides.side2.append(target) 0(n)
    else: 0(1)
```

0(n) linear complexity

```
        #moves the animal from side 2 (comment on the code)
        sides.side2.remove(target) 0(n)
        sides.side1.append(target)  0(n)
  else: 0(1)
```

0(n) linear complexity

#if the user has not chose to move only student then it will go down this path as it does contain the additional append and remove of student to account for the student being the only one able to move the raft (comment on the code)

Constant complexity 0(1)

```
    if location == "side1": 0(1)
        #this if statement checks to make sure the student is on the same side as the animal
being moved (comment on the code)
```

Constant complexity 0(1)

```
        if "Student" not in sides.side1:
```

0(n) linear complexity

```
 #displays error (comment on the code)
            print("Student is not on side 1 so they cannot move the animal ")
        else: 0(1)
            #moves the animal from side 1 (comment on the code)
            sides.side1.remove("Student") 0(n)
            sides.side2.append("Student") 0(n)
            sides.side1.remove(target) 0(n)
            sides.side2.append(target) 0(n)
```

0(n) linear complexity

```
    else:
        #if statement to check if the student is on the same side as the animal being moved
```

Constant complexity 0(1)

```
 (comment on the code)
        if "Student" not in sides.side2:   0(1)
            print("Student is not on side 2 so they cannot move ") 0(1)
```

Constant complexity 0(1)

```
        else:
            #moves the animal from side 2  (comment on the code)
            sides.side2.remove(target) 0(n)
            sides.side1.append(target) 0(n)
            sides.side2.remove("Student") 0(n)
            sides.side1.append("Student") 0(n)
```

0(n) linear complexity

```
def checkForWin():
#check if all entities are on side 2 (comment on the code)
```
Constant complexity 0(1)

```
    if "Student" in sides.side2 and "Dog" in sides.side2 and "Cat" in sides.side2 and "Hamster" in
sides.side2: 0(n2)
```
0(n) linear complexity
```
        #win is set to true if they are (comment on the code)
        win = True 0(1)
```
Constant complexity 0(1)
```
    else:
        #win is set to false if they are not (comment on the code)
        win = False 0(1)
```
Constant complexity 0(1)
```
    #return boolean win to main (comment on the code)
    return win
```
There is constant time for checking conditions. The space is constant at 0(1)

```
def checkForConflicts():  0(1)
    #series of checks to make sure no conflicts occur during movement (comment on the code)
    if "Dog" in sides.side1 and "Cat" in sides.side1 and "Student" not in sides.side1:
      input("Dog attacked Cat on side1")  0(n)
```
Linear complexity
```
    elif  "Dog" in sides.side2 and "Cat" in sides.side2 and "Student" not in sides.side2:
      input("Dog attacked Cat on side2")  0(n)
```
Linear complexity

```
    elif "Cat" in sides.side1 and "Hamster" in sides.side1 and "Student" not in sides.side1:
      input("Cat attacked Hamster on side1")  0(n)
```
Linear complexity

```
    elif "Cat" in sides.side2 and "Hamster" in sides.side2 and "Student" not in sides.side2:
      input("Cat attacked Hamster on side2") 0(n)
```
Linear complexity

The time is constant as it checks conditions and prints messages. The space is also constant at 0(1)

```
def checkTargetAndLocation(): 0(1)
    target = False  0(1)
    while target == False: 0(n)
        userTarget = input("What target would you like to move? ") 0(n)
        #check to check if the target the user has chosen is in either side (comment on the code)
        if userTarget in sides.side1 or userTarget in sides.side2: 0(n)
```
Linear complexity 0(n)
```
          target = True 0(1)
          location = False 0(1)
          while location == False: 0(1)
```

```
        userLocation = input("From which side? side1 or side2? ") 0(1)
```
Constant complexity 0(1)
```
        #check to check if the users picked side is an actual side (comment on the code)
        if userLocation == "side1" or userLocation == "side2": 0(1)
```
Constant complexity 0(1)
```
          #check to check if the side the user has picked contains the target they have picked (comment on the code)
          if userTarget not in getattr(sides, userLocation):
            location = False 0(1)
            print("That target is not on that side")
          else:
```
Constant complexity
```
            #if all checks are true then the animal is moved (comment on the code)
            location = True
```
Constant complexity
```
moveAnimal(userTarget, userLocation)
            win = False 0(1)
        else:
          #display if a correct side has not been picked (comment on the code)
          print("That is not a side")  0(1)
```
constant complexity
```
    else:
      #display if a correct target has not been picked (comment on the code)
      print("Sorry that is not a target") 0(1)
```
The time complexity here depends on the users input choices, so it is different depending on what the user decided to do. The space complexity remains at 0(1)

```
main()
```
The overall time complexity is dependent on user input and if they choose correctly or not. The space remains constant at 0(1)
The code uses a large amount of constant time and space complexity. It mainly comes down to the users interactions and if they complete the puzzle correctly.

<u>Overall of the complexity</u>

0(n)

Team Evaluation

Hitesh Lad - 23011542 - Complexity
During the assignment, Hitesh has worked well with the other members of the team to get their respective tasks completed to the best of their ability. Once the Python code was ready, Hitesh worked to get the complexity and justification complete and performed several checks to ensure that the calculations were correct. His contribution has been invaluable to the team and helped to get us to a finished state.

Connor Hutchison - 23029981-
Pseudocode
As required Connor worked exceptionally well as a group as he was able to provide his thoughts on each task to be able to help his peers complete the tasks to the best of their ability. The task which Connor completed was

pseudocode which he completed to an excellent standard. His help and efforts were appreciated as he was able to help us all progress in this assignment.

Mohammed Ismail - 23011237 - Python
As required, Ismail helped his team, provided great input and developed and wrote a well written program. He was greatly appreciated and provided us with a plan to help make the team perform to its potential. His task was to develop and write up a solution for problem 1 in Python which was accompanied alongside running the test cases for the code and completing the ChatGPT analysis. Ismail was an important member with his contribution and development of the python code.

Haris Kadu – 22009453 - Flowchart
During this assignment, Haris performed efficiently and effectively when needed, providing full support to his peers and completing his task to a high standard. The task given to Haris was to create a flowchart which represents the python code a member of his team wrote. Haris performance was appreciated and allowed the teams to perform at a high efficacy.