**Assignment 5**

**Question 1**

Name 5 sorting algorithms, also write their time complexities (best, average, worst).

**Answer 1**

Sorting algorithms are:

1. Selection sort

Best: Ω(n^2)

Average: ϴ(n^2)

Worst: O(n^2)

1. Bubble sort

Best: Ω(n)

Average: ϴ(n^2)

Worst: O(n^2)

1. Insertion sort

Best: Ω(n)

Average: ϴ(n^2)

Worst: O(n^2)

1. Quick sort

Best: Ω(n log(n))

Average: ϴ(n log(n))

Worst: O(n^2)

1. Heap sort

Best: Ω(n log(n))

Average: ϴ(n log(n))

Worst: O(n log(n))

**Question 2**

Implement selection sort algorithm using Python.

**Answer 2**

def selectionSort( itemsList ):

n = len( itemsList )

for i in range( n - 1 ):

minValueIndex = i

for j in range( i + 1, n ):

if itemsList[j] < itemsList[minValueIndex] :

minValueIndex = j

if minValueIndex != i :

temp = itemsList[i]

itemsList[i] = itemsList[minValueIndex]

itemsList[minValueIndex] = temp

return itemsList

el = [21,6,9,33,3]

print(selectionSort(el))

**Question 3**

Implement pop operation of the stack.

**Answer 3**

stack = [ ]

stack.append('a')

stack.append('b')

stack.append('c')

print('Initial stack')

print(stack)

print('\nElements poped from stack:')

print(stack.pop())

print(stack.pop())

print(stack.pop())

print('\nStack after elements are poped:')

print(stack)

**Question 4**

Implement dequeue operation of the queue

**Answer 4**

class Queue:

def \_\_init\_\_(self):

self.head = None

self.tail = None

def enqueue(self, element):

if self.head is None:

self.head = self.tail = Node(element)

else:

node = Node(element)

self.tail.next\_node = node

self.tail = node

def dequeue(self):

element = self.head.element

if self.tail == self.head:

self.tail = self.head = None

else:

self.head = self.head.next\_node

return element