***Total Time Taken: 02:44:01.31 (hour:min:sec:millisec)***

|  |  |
| --- | --- |
| Candidate name: | Adler Chua Yu Cheng |
| Centre number: |  |
| Index number: |  |
| Programming language used: | Python 3 |
| **Question 1**  **Evidence 1** | |
| *Paste program code here*  def display\_menu():  print("\n+++++++++++++++++++++++"\  "\n1. Exact match"\  "\n2. Start of term"\  "\n3. Within term"\  "\n++++++++++++++++++")  def get\_choice():  while True:  choice = input("Choice ?")  if choice == 'XXX':  return choice  if choice.isdigit():  choice = int(choice)  if choice in range(4):  return choice  print("Invalid input!"\  " Please the number corresponding to your desired option.")      def read\_file():  f = open("JARGON.TXT")  temp = f.read().split('\n')[:-1]  f.close()  return temp  def menu():  contents = read\_file()  while True:  display\_menu()  choice = get\_choice()  if choice == 'XXX':  break  term = input("Term?")  result = []  if choice == 1:  for i in range(len(contents)):  if contents[i] == term:  result.append(contents[i])  elif choice == 2:  for i in range(len(contents)):  if contents[i][:len(term)] == term:  result.append(contents[i])  elif choice == 3:  for i in range(len(contents)):  for j in range(len(contents)):  if contents[i][j: j + len(term)] == term and\  (not search(result, contents[i])):  result.append(contents[i])  for i in result:  print(i)  print("There were " + str(len(result)) + " matching term(s)")  def search(array, target):  for i in range(len(array)):  if array[i] == target:  return True  return False  menu() | |
| **Evidence 2** | |
| *Test data summary*  No repetition  Capitalisation sensitive  Not found  Does not overlap with other functions  *Paste screenshot here*  Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-01 at 6.10.01 PM.pngMacintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-01 at 6.11.05 PM.pngMacintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-01 at 6.10.56 PM.pngMacintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-01 at 6.10.35 PM.png | |
| **Question 2**  **Evidence 3** | |
| **A** Low > High  **B** (Low + High) MOD 2  **C** RETURN BinarySearch(ThisArray, FindValue, Middle + 1, High) | |
| **Evidence 4** | |
| *Paste program code here*  def binary\_search(this\_array, find\_value, low, high): #returns INT  if low > high: #A  return -1 #not found  else:  #calculate new middle value  middle = (low + high) // 2 #B  if this\_array[middle] > find\_value:  return binary\_search(this\_array, find\_value, low, middle - 1)  elif this\_array[middle] < find\_value:  return binary\_search(this\_array, find\_value, middle + 1, high) #C  else:  return middle  def initialise\_animals():  MyAnimal = [None for i in range(33)]  MyAnimal[0]="aardvark"  MyAnimal[1]="ant"  MyAnimal[2]="antelope"  MyAnimal[3]="bat"  MyAnimal[4]="boa constrictor"  MyAnimal[5]="camel"  MyAnimal[6]="cat"  MyAnimal[7]="cheetah"  MyAnimal[8]="dog"  MyAnimal[9]="donkey"  MyAnimal[10]="duck"  MyAnimal[11]="elephant"  MyAnimal[12]="frog"  MyAnimal[13]="giraffe"  MyAnimal[14]="hare"  MyAnimal[15]="horse"  MyAnimal[16]="iguana"  MyAnimal[17]="jackass"  MyAnimal[18]="jaguar"  MyAnimal[19]="leopard"  MyAnimal[20]="lion"  MyAnimal[21]="llama"  MyAnimal[22]="mouse"  MyAnimal[23]="ostrich"  MyAnimal[24]="panther"  MyAnimal[25]="parrot"  MyAnimal[26]="rhinoceros"  MyAnimal[27]="seahorse"  MyAnimal[28]="seal"  MyAnimal[29]="spider"  MyAnimal[30]="turtle"  MyAnimal[31]="whale"  MyAnimal[32]="zebra"  return MyAnimal  def find\_animal():  my\_animal = initialise\_animals()  name = input("Animal name: ")  index = binary\_search(my\_animal, name, 0, len(my\_animal) - 1)  if index == -1:  print("Animal not found.")  else:  print("Animal found at index " + str(index))  find\_animal() | |
| **Evidence 5** | |
| *Paste screenshot here*  Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-01 at 6.25.02 PM.png | |
| **Evidence 6** | |
| *Paste program code here*  def binary\_search\_amended(this\_array, find\_value, low, high, calls = 0):  #returns INT  if low > high: #A  return -1, calls #not found  else:  #calculate new middle value  middle = (low + high) // 2 #B  if this\_array[middle] > find\_value:  return binary\_search\_amended(this\_array, find\_value, low, \  middle - 1, calls + 1)  elif this\_array[middle] < find\_value:  return binary\_search\_amended(this\_array, find\_value, middle + 1, \  high, calls + 1) #C  else:  return middle, calls    def find\_animal\_amended():  my\_animal = initialise\_animals()  name = input("Animal name: ")  index, calls = binary\_search\_amended(my\_animal, name, 0, len(my\_animal) - 1)  if index == -1:  print("Animal not found with " + str(calls) + " function calls")  else:  print("Animal found at index " + str(index) + " with " + str(calls)\  + " function calls")  find\_animal\_amended() | |
| **Evidence 7** | |
| *Paste screenshot here X 2*  **Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-01 at 6.33.58 PM.png**  **Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-01 at 6.33.50 PM.png** | |
| **Question 3**  **Evidence 8** | |
| *Paste program code here*  def menu(linked\_list):  while True:  choice = get\_choice()  if choice == 5:  break  elif choice == 1:  linked\_list.add\_node()  elif choice == 2:  linked\_list.traversal()  elif choice == 3:  linked\_list.display\_linked\_list  def get\_choice():  while True:  choice = input("\n1. Add an item"\  "\n2. Traverse the linked list of used nodes"\  " and output the data values"\  "\n3. Output all pointers and data values"\  "\n5. Exit\n")  if choice.isdigit():  choice = int(choice)  if choice in range(1,4) or choice == 5:  return choice  print("Invalid input! Enter a digit"\  " corresponding to the desired option.") | |
| **Evidence 9** | |
| *Paste program code here*  class ListNode():  def \_\_init\_\_(self, data\_value):  self.\_data\_value = data\_value  self.\_pointer\_value = 0  def get\_data\_value(self):  return self.\_data\_value  def set\_data\_value(self, new\_data\_value):  self.\_data\_value = new\_data\_value  def get\_pointer\_value(self):  return self.\_pointer\_value  def set\_pointer\_value(self, new\_pointer\_value):  self.\_pointer\_value = new\_pointer\_value  class LinkedList():  def \_\_init\_\_(self):  self.\_node = [None]  for i in range(1, 30):  temp = ListNode('')  temp.set\_pointer\_value(i + 1)  self.\_node.append(temp)  self.\_node.append(ListNode(''))  self.\_start = 0  self.\_next\_free = 1  def display\_linked\_list(self):  print("Start: " + str(self.\_start))  print("Next Free: " + str(self.\_next\_free))  for i in range(1, len(self.\_node)):  print("Index: {:<20}Data Value: {:<20}Pointer Value: {}"\  .format(str(i), str(self.\_node[i].get\_data\_value()),\  str(self.\_node[i].get\_pointer\_value())))  def is\_empty(self):  return self.\_start == 0 | |
| **Evidence 10** | |
| *Paste screenshot here*  **Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-02 at 10.36.59 AM.png** | |
| **Evidence 11** | |
| *Paste program code here*    def add\_node(self):  if self.is\_full():  print("Linked list is full.")  return False  new\_item = input("New Data Value: ")  self.\_node[self.\_next\_free].set\_data\_value(new\_item)  if self.\_start == 0:  self.\_start = self.\_next\_free  temp = self.\_node[self.\_next\_free].get\_pointer\_value()  self.\_node[self.\_next\_free].set\_pointer\_value(0)  self.\_next\_free = temp  else:  #traverse the list ñ at Start to find  #the position at which to insert the new item  temp = self.\_node[self.\_next\_free].get\_pointer\_value()  if new\_item < self.\_node[self.\_start].get\_data\_value():  #new item will become the start of the list  self.\_node[self.\_next\_free].set\_pointer\_value(self.\_start)  self.\_start = self.\_next\_free  self.\_next\_free = temp  else:  #the new item is not at the start of the list . . .  previous = 0  current = self.\_start  found = False  while not found and current != 0:  if new\_item <= self.\_node[current].get\_data\_value():  self.\_node[previous].set\_pointer\_value(self.\_next\_free)  self.\_node[self.\_next\_free].set\_pointer\_value(current)  self.\_next\_free = temp  found = True  else:  #move on the next node  previous = current  current = self.\_node[current].get\_pointer\_value()  if current == 0:  self.\_node[previous].set\_pointer\_value(self.\_next\_free)  self.\_node[self.\_next\_free].set\_pointer\_value(0)  self.\_next\_free = temp  def is\_full(self):  return self.\_next\_free == 0 | |
| **Evidence 12** | |
| *Paste screenshot here*  **Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-02 at 10.53.30 AM.png** | |
| **Evidence 13** | |
| *Paste program code here*  def traversal(self):  self.traversal\_in\_order(self.\_start)    def traversal\_in\_order(self, index):  if index != 0:  print(self.\_node[index].get\_data\_value())  #follow the pointer to the next data item in the linked list  self.traversal\_in\_order(self.\_node[index].get\_pointer\_value()) | |
| **Evidence 14** | |
| *Paste screenshot here*  **Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-02 at 10.59.55 AM.png** | |
| **Evidence 15** | |
| *Paste program code here*  def menu(linked\_list):  while True:  choice = get\_choice()  if choice == 5:  break  elif choice == 1:  linked\_list.add\_node()  elif choice == 2:  linked\_list.traversal()  elif choice == 3:  linked\_list.display\_linked\_list()  elif choice == 4:  linked\_list.reverse\_traversal()  def get\_choice():  while True:  choice = input("\n1. Add an item"\  "\n2. Traverse the linked list of used nodes"\  " and output the data values"\  "\n3. Output all pointers and data values"\  "\n4. Traverse the linked list of used nodes"\  " and output the data values in reverse order"\  "\n5. Exit\n")  if choice.isdigit():  choice = int(choice)  if choice in range(1,6):  return choice  print("Invalid input! Enter a digit"\  " corresponding to the desired option.")  class ListNode():  def \_\_init\_\_(self, data\_value):  self.\_data\_value = data\_value  self.\_pointer\_value = 0  def get\_data\_value(self):  return self.\_data\_value  def set\_data\_value(self, new\_data\_value):  self.\_data\_value = new\_data\_value  def get\_pointer\_value(self):  return self.\_pointer\_value  def set\_pointer\_value(self, new\_pointer\_value):  self.\_pointer\_value = new\_pointer\_value  class LinkedList():  def \_\_init\_\_(self):  self.\_node = [None]  for i in range(1, 30):  temp = ListNode('')  temp.set\_pointer\_value(i + 1)  self.\_node.append(temp)  self.\_node.append(ListNode(''))  self.\_start = 0  self.\_next\_free = 1    def add\_node(self):  if self.is\_full():  print("Linked list is full.")  return False  new\_item = input("New Data Value: ")  self.\_node[self.\_next\_free].set\_data\_value(new\_item)  if self.\_start == 0:  self.\_start = self.\_next\_free  temp = self.\_node[self.\_next\_free].get\_pointer\_value()  self.\_node[self.\_next\_free].set\_pointer\_value(0)  self.\_next\_free = temp  else:  #traverse the list ñ at Start to find  #the position at which to insert the new item  temp = self.\_node[self.\_next\_free].get\_pointer\_value()  if new\_item < self.\_node[self.\_start].get\_data\_value():  #new item will become the start of the list  self.\_node[self.\_next\_free].set\_pointer\_value(self.\_start)  self.\_start = self.\_next\_free  self.\_next\_free = temp  else:  #the new item is not at the start of the list . . .  previous = 0  current = self.\_start  found = False  while not found and current != 0:  if new\_item <= self.\_node[current].get\_data\_value():  self.\_node[previous].set\_pointer\_value(self.\_next\_free)  self.\_node[self.\_next\_free].set\_pointer\_value(current)  self.\_next\_free = temp  found = True  else:  #move on the next node  previous = current  current = self.\_node[current].get\_pointer\_value()  if current == 0:  self.\_node[previous].set\_pointer\_value(self.\_next\_free)  self.\_node[self.\_next\_free].set\_pointer\_value(0)  self.\_next\_free = temp  def display\_linked\_list(self):  print("Start: " + str(self.\_start))  print("Next Free: " + str(self.\_next\_free))  for i in range(1, len(self.\_node)):  print("Index: {:<20}Data Value: {:<20}Pointer Value: {}"\  .format(str(i), str(self.\_node[i].get\_data\_value()),\  str(self.\_node[i].get\_pointer\_value())))    def is\_empty(self):  return self.\_start == 0  def is\_full(self):  return self.\_next\_free == 0  def traversal(self):  self.traversal\_in\_order(self.\_start)    def traversal\_in\_order(self, index):  if index != 0:  print(self.\_node[index].get\_data\_value())  #follow the pointer to the next data item in the linked list  self.traversal\_in\_order(self.\_node[index].get\_pointer\_value())    def reverse\_traversal(self):  self.traversal\_in\_reverse\_order(self.\_start)  def traversal\_in\_reverse\_order(self, index):  if index != 0:  #follow the pointer to the next data item in the linked list  self.traversal\_in\_reverse\_order(\  self.\_node[index].get\_pointer\_value())  print(self.\_node[index].get\_data\_value())    menu(LinkedList()) | |
| **Evidence 16** | |
| *Paste screenshot here*  **Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-02 at 11.03.53 AM.png** | |
| **Question 4**  **Evidence 17** | |
| *Program design statements*  #4.1  #grid will be an class (class Grid) using an array of rows,  #which is represented by a class data structure (class Row),  #which takes in parameter of its row number,  #and is an array consisting of nodes (class Node),  #that has parameters: data (ascii code); x-coordinate(int); y-coordinate(int),  #with get, set, and print methods | |
| **Evidence 18** | |
| *Paste program code here*  class Grid():  def \_\_init\_\_(self):  self.\_grid = [None] + [Row(i) for i in range(1, 9)]  def display(self):  for i in range(1, len(self.\_grid)):  self.\_grid[i].display()  print()  def reset(self):  self.\_\_init\_\_()  def throw\_stone(self):  while True:  x = input('X coordinate <1 to 15>? ')  if x.isdigit():  x = int(x)  if x in range(1, 16):  break  print("Invalid input! Enter a valid X coordinate.")  while True:  y = input('y coordinate <1 to 8>? ')  if y.isdigit():  y = int(y)  if y in range(1, 9):  break  print("Invalid input! Enter a valid Y coordinate.")  self.\_grid[y].get\_node(x).set\_data(83)  self.display()  class Row():  def \_\_init\_\_(self, row\_num):  self.\_row\_num = row\_num  self.\_row = [None] + [Node(i, self.\_row\_num) for i in range(1, 16)]  def display(self):  for i in range(1, len(self.\_row)):  self.\_row[i].display()  def reset(self):  self.\_\_init\_\_(row\_num)  def get\_node(self, x): #where x is the node's x-coordinate  return self.\_row[x]    class Node():  #node's data is represented by ascii code values of:  #. for one sq m of water (i.e.: nothing there)  #S for stone impact position;  #F for fish  #P for pellet  #H for happy (fed) fish  def \_\_init\_\_(self, x, y, data = 46):  #where x is the x-coordinate and y is the y-coordinate  self.\_data = data  self.\_x = x  self.\_y = y  def get\_data(self):  return self.\_data  def set\_data(self, new\_data):  self.\_data = new\_data  def get\_x(self):  return self.\_x  def set\_x(self, new\_x):  self.\_x = new\_x  def get\_y(self):  return self.\_y  def set\_y(self, new\_y):  self.\_y = new\_y  def display(self):  print(chr(self.\_data), end = '')  grid = Grid()  grid.throw\_stone() | |
| **Evidence 19** | |
| *Paste screenshot here*  **Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-03 at 1.01.37 AM.png** | |
| **Evidence 20** | |
| *Paste program code here*  import random  class Grid():  def \_\_init\_\_(self):  self.\_grid = [Row(i) for i in range(1, 9)]  self.\_fish\_count = 0  def display(self):  for i in range(len(self.\_grid)):  self.\_grid[i].display()  print()  def reset(self):  self.\_\_init\_\_()  def throw\_stone(self):  while True:  x = input('X coordinate <1 to 15>? ')  if x.isdigit():  x = int(x)  if x in range(1, 16):  break  print("Invalid input! Enter a valid X coordinate.")  while True:  y = input('y coordinate <1 to 8>? ')  if y.isdigit():  y = int(y)  if y in range(1, 9):  break  print("Invalid input! Enter a valid Y coordinate.")  self.\_grid[y].get\_node(x).set\_data(83)  self.display()  def add\_three\_fish(self):  while self.\_fish\_count != 3:  self.add\_fish()  self.display()    def add\_fish(self):  if self.\_fish\_count < 3:  x = random.randint(1, 15)  y = random.randint(1, 8)  self.\_grid[y].get\_node(x).set\_data(70)  self.\_fish\_count += 1  class Row():  def \_\_init\_\_(self, row\_num):  self.\_row\_num = row\_num  self.\_row = [Node(i, self.\_row\_num) for i in range(1, 16)]  def display(self):  for i in range(len(self.\_row)):  self.\_row[i].display()  def reset(self):  self.\_\_init\_\_(row\_num)  def get\_node(self, x): #where x is the node's x-coordinate  return self.\_row[x]  grid = Grid()  grid.add\_three\_fish() | |
| **Evidence 21** | |
| *Paste screenshot here*  *Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-03 at 1.01.51 AM.png* | |
| **Evidence 22** | |
| *Paste program code here*  class Grid():  def \_\_init\_\_(self):  self.\_grid = [None] + [Row(i) for i in range(1, 9)]  self.\_fish\_count = 0  def display(self):  for i in range(1, len(self.\_grid)):  self.\_grid[i].display()  print()  def reset(self):  self.\_\_init\_\_()  def \_get\_coordinates(self):  while True:  x = input('X coordinate <1 to 15>? ')  if x.isdigit():  x = int(x)  if x in range(1, 16):  break  print("Invalid input! Enter a valid X coordinate.")  while True:  y = input('y coordinate <1 to 8>? ')  if y.isdigit():  y = int(y)  if y in range(1, 9):  break  print("Invalid input! Enter a valid Y coordinate.")  return x, y    def throw\_stone(self):  x, y = self.\_get\_coordinates()  self.\_grid[y].get\_node(x).set\_data(83)  self.display()    def add\_three\_fish(self):  while self.\_fish\_count != 3:  self.add\_fish()  self.display()    def add\_fish(self):  if self.\_fish\_count < 3:  x = random.randint(1, 15)  y = random.randint(1, 8)  self.\_grid[y].get\_node(x).set\_data(70)  self.\_fish\_count += 1  def throw\_pellet(self):  x, y = self.\_get\_coordinates()  temp\_node = self.\_grid[y].get\_node(x)  if temp\_node.get\_data() == 70:  temp\_node.set\_data(72)  elif temp\_node.get\_data() == 46:  temp\_node.set\_data(80)  #else:  #raise IndexError  self.display() | |
| **Evidence 23** | |
| *Paste screenshot here X 2*  One throw which did not feed a fish  *Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-03 at 1.09.24 AM.png*    A second throw where a fish was fed  *Macintosh HD:Users:AdlerChuaYuCheng:Desktop:Screen Shot 2018-08-03 at 1.07.41 AM.png* | |