ITDPA2-B44 Assignment:

Index:

Section A: Question 1	p.3
Section A: Question 2	p.5
Section A: Question 3	p.10
Bibliography	p.16

Section A:

Question 1:

Input:

```
def Main(): #Create the Main function to execute the whole program

    class PatientFileReader: #Create the PatientFileReader class with its own functions
        def __init__(self):
            self.file = None #Create self.file that will be used to get data throughout the class

        def Open(self): #Create the Open function that will open the text file for use
            self.file = open("C:\\Users\ether\Desktop\Patient_Data.txt","r+")

        def Close(self): #Create the Close function that will close the text file
            self.file.close()

        def GetAll(self): #Create the GetAll function that will store the text file in a list format
            Lines = self.file.readlines()

        def GetData(self): #Create the GetData function that will read the lines of the text file
            X = 0  #counter
            while X != 5: #While not the end of the text file
                print (self.file.readline()) #prints each line of the text file
                X = X + 1 #Increment X


    FileReader = PatientFileReader() #Create the class variable


    class PatientData: #Create the PatientData class with its own functions
        def __init__(self):
            self.Data = None

        def PrintData(self): #Create the PrintData funtion to print all of the Patient's Data
            print('Patient Data')
            print('')
            print('Patient no.', 'Surname', 'Name', 'Age Group', 'Sex', 'Weight(Kg)')
            print('-'*45)
            FileReader.Open() #Opens the Text file
            Text = FileReader.GetData() #Receives all the information from the text file and stores it in the Text variable
            FileReader.Close() #Closes the Text file
            print(Text) #Prints everything from the text file
            print('Number of Patients: 5')

    PatientDataReader = PatientData()#Create the class variable
    PatientDataReader.PrintData() #Call the PrintData function to print everything
Main() #Executes the whole program
```

Output:

```
Patient Data

Patient no. Surname Name Age Group Sex Weight(Kg)
-----------------------------------------
10005, Linda, Khumalo, Young Adult, F, 56.40

10012, Mokoena, Mkhize, School Age Child, M, 9.82

10015, Dlamini, Nkosi, Infant, M, 5.60

10100, Sithole, Mahlangu, Older Adult, M, 91.77

10132, Ntombi, Zulu, Pre-school child, F, 10.61

None
Number of Patients: 5
```

Text file input:

```
10005, Linda, Khumalo, Young Adult, F, 56.40
10012, Mokoena, Mkhize, School Age Child, M, 9.82
10015, Dlamini, Nkosi, Infant, M, 5.60
10100, Sithole, Mahlangu, Older Adult, M, 91.77
10132, Ntombi, Zulu, Pre-school child, F, 10.61
```

Question 2:

2.1

Input:

```
from array import * #import array functions to use it

x = 0 #create two counters x and y
y = 0
array1 = array('i',[0,1,2,3,4,5]) #first array and its 6 placeholders
array2 = array('i', [0,1,2,3]) #second array and its 4 placeholders

while x != 6: #While 6 inputs are not entered, the loop will continue
    Input1 = int(input("Please enter any 6 numbers")) #prompts user to enter any number between 1 and 12
    if Input1 >= 1 and Input1 <= 12: #checks if the entered value is between 1 and 12
        array1[x] = Input1 #adds the input to the array1
        x = x +1 #Moves x to the next element in the array1
    else:
        print('Wrong, Please enter another number between 1 and 12') #The user must enter a correct number
for f in array1:
    if (f%2) == 0: #Checks if f is an even number
        print(f) #Prints the even number
while y != 4: #While 4 inputs are not entered, the loop will continue
    Input2 = int(input("Please enter any 4 numbers")) #prompts user to enter any number between 20 and 40
    if Input2 >= 20 and Input2 <= 40: #checks if the entered value is between 20 and 40
        array2[y] = Input2 #adds the input to the array2
        y = y +1 #Moves x to the next element in the array2
    else:
        print('Wrong, Please enter another number between 20 and 40') #The user must enter a correct number
for f in array2:
    if (f%2) != 0:#Checks if f is an odd number
        print(f) #Prints the odd number
```

Output:

```
Please enter any 6 numbers12
Please enter any 6 numbers5
Please enter any 6 numbers9
Please enter any 6 numbers8
Please enter any 6 numbers6
Please enter any 6 numbers4
12
8
6
4
Please enter any 4 numbers40
Please enter any 4 numbers22
Please enter any 4 numbers35
Please enter any 4 numbers30
35
```

All Even numbers displays for array1, and all odd numbers displays for array2

2.2

Input:

Function to sort the merged arrays

```
def MergeSort(Unsorted): #create a function to sort the merged array
    x = 0 #Create counters
    y = 0
    z = 0

    if len(Unsorted) > 1: #gets the values of the merged array
        middle = len(Unsorted)//2
        Left = Unsorted[:middle]
        Right = Unsorted[middle:]

        MergeSort(Left)
        MergeSort(Right)

        while x < len(Left) and y < len(Right): #while x is smaller than the length of the left side and y smaller than right

            if Left[x] < Right[y]: #changes the position of the smallest value to the left side
                Unsorted[z] = Left[x]
                x = x + 1
            else:
                Unsorted[z] = Right[y] #changes the position of the highest value to the right side
                y = y + 1
            z = z + 1

        while x < len(Left): #changes the position the where x is
            Unsorted[z] = Left[x]
            x = x + 1
            z = z + 1

        while y < len(Right): #changes the position the where y is
            Unsorted[z] = Right[y]
            y = y + 1
            z = z + 1
```

Function to merge the two arrays

```
def Merge(array1, array2): #create a function to merge an array
    e = 0 #create counters for the two arrays
    f = 0

    Lengtharr1 = len(array1) #get the legths of the two arrays
    Lengtharr2 = len(array2)

    Newarray = [] #create a new array for the merged array

    while ((e < Lengtharr1) and (f < Lengtharr2)): #while both the counters are smaller than both array's lenghts
        if array1[e] < array2[f]: #if the element value of the first array is smaller than the second array
            Newarray.append(array1[e]) #adds the value of the current element in the 1st array to the newarray elements
            e = e + 1 #increment e
        else:
            Newarray.append(array2[f])  #else the value of the current element in the 2nd array gets added to the newarray
            f = f + 1 #increment f
    while (e < Lengtharr1): #while the length of the first array is still bigger than e
        Newarray.append(array1[e]) #adds the current value of e in array1 to Newarray
        e = e + 1 #increment e

    while (f < Lengtharr2): #while the length of the second array is still bigger than f
        Newarray.append(array2[f]) #adds the current value of f in array2 to Newarray
        f = f + 1 #increment f
    return Newarray #returns the Newarray and the value of its elements

Newarray = Merge(array1,array2) #calls the merged array
print(Newarray) #prints the merged array
```

Output:

```
Please enter any 6 numbers12
Please enter any 6 numbers9
Please enter any 6 numbers4
Please enter any 6 numbers8
Please enter any 6 numbers5
Please enter any 6 numbers3
12
4
8
Please enter any 4 numbers35
Please enter any 4 numbers22
Please enter any 4 numbers24
Please enter any 4 numbers31
35
31
[12, 9, 4, 8, 5, 3, 35, 22, 24, 31]
```

2.3 a:

Input of the linked list, following the other codes:

```
class Node: #Create the class for the linked list
    def __init__(self, data):
        self.data = data #will carry the data to be passed through the linked list
        self.next = None #will switch to the next value in the linked list

def Insert(root, number): #create function to insert values into the linked list
    temp = Node(number) #placeholder for the values

    if (root == None): #if root does not have a value
        root = temp  #it gains the value of temp
    else:
        Printer = root #Printer gets the value of root
        while (Printer.next != None): #if the next value of Printer is not equal to none
            Printer = Printer.next #printer will get the next value
        Printer.next = temp #temp passes on the next value aquired to Printer
    return root #returns the value of root, which will have the value of printer

def display(root):#create function to display values of the linked list
    while(root != None): #if there is a value in root
        print(root.data, end = "-->") #the value of root will be printed, which an arrow next to it
        root = root.next #gets the next value of the linked list

def MergeToList(Newarray, n): #create a function that will covert the array into the linked list
    root = None #will be used to insert values into the linked list
    for X in range(0, n, 1): #X continues to loop through the array
        root = Insert(root, Newarray[X]) #Enters the value obtained by X from the array, into the linked list
    return root #returns the completed linked list

n = len(Newarray) #n will be the length of the array
root = MergeToList(Newarray, n) #calls the function to convert the array into a linked list
print("Linked List in same order of merged sorted arrays:")
display(root) #displays the created linked list
```

Output:

```
Please enter any 6 numbers10
Please enter any 6 numbers2
Please enter any 6 numbers5
Please enter any 6 numbers6
Please enter any 6 numbers8
Please enter any 6 numbers9
10
2
6
8
Please enter any 4 numbers20
Please enter any 4 numbers22
Please enter any 4 numbers32
Please enter any 4 numbers40
Linked List in same order of merged sorted arrays:
10-->2-->5-->6-->8-->9-->20-->22-->32-->40-->
```

2.3 b:

```
class Node: #Create the class for the linked list
    def __init__(self, data):
        self.data = data #will carry the data to be passed through the linked list
        self.next = None #will switch to the next value in the linked list

def Insert(root, number): #create function to insert values into the linked list
    temp = Node(number) #placeholder for the values

    if (root == None): #if root does not have a value
        root = temp  #it gains the value of temp
    else:
        Printer = root #Printer gets the value of root
        while (Printer.next != None): #if the next value of Printer is not equal to none
            Printer = Printer.next #printer will get the next value
        Printer.next = temp #temp passes on the next value aquired to Printer
    return root #returns the value of root, which will have the value of printer

def display(root):#create function to display values of the linked list
    while(root != None): #if there is a value in root
        print(root.data, end = "-->") #the value of root will be printed, which an arrow next to it
        root = root.next #gets the next value of the linked list

def MergeToList(Newarray, n): #create a function that will covert the array into the linked list
    root = None #will be used to insert values into the linked list
    for X in range(0, n, 1): #X continues to loop through the array
        root = Insert(root, Newarray[X]) #Enters the value obtained by X from the array, into the linked list
    return root #returns the completed linked list

n = len(Newarray) #n will be the length of the array
root = MergeToList(Newarray, n) #calls the function to convert the array into a linked list
print("Linked List in reverse order of merged sorted arrays:")
display(root) #displays the created linked list
```

```
Please enter any 6 numbers4
Please enter any 6 numbers10
Please enter any 6 numbers12
Please enter any 6 numbers5
Please enter any 6 numbers8
Please enter any 6 numbers6
4
10
12
8
6
Please enter any 4 numbers22
Please enter any 4 numbers31
Please enter any 4 numbers40
Please enter any 4 numbers35
31
35
Linked List in reverse order of merged sorted arrays:
35-->40-->31-->22-->6-->8-->5-->12-->10-->4-->
```

2.3 c:

Input:

```
n = len(Newarray) #n will be the length of the array
root = MergeToList(Newarray, n) #calls the function to convert the array into a linked list
Value45 = 45 #Give the Value45 variable the value of 45
root = Insert(root,Value45) #Inserts the value of 45 into the created link list
print("Linked List in reverse order of merged sorted arrays after the value of 45 is added:")
display(root) #displays the created linked list
```

```
Please enter any 6 numbers10
Please enter any 6 numbers12
Please enter any 6 numbers5
Please enter any 6 numbers6
Please enter any 6 numbers8
Please enter any 6 numbers9
10
12
6
8
Please enter any 4 numbers22
Please enter any 4 numbers31
Please enter any 4 numbers40
Please enter any 4 numbers22
31
Linked List in reverse order of merged sorted arrays after the value of 45 is added:
22-->40-->31-->22-->9-->8-->6-->5-->12-->10-->45-->
```

Question 3:

3.1

Input for Shop Tree:

```
class Shop(object): #Create the class for the 1st Tree
    def __init__(self, data):
        self.left = None #Create a variable for the left node of the tree
        self.right = None #Create a variable for the right node of the tree
        self.data = data #Variable for passing and entering data

    def Insert(self, data): #Function to insert data into the tree
        if self.data == None: #If there is no value in data
            self.data = data  #Self.data gets a value from data
        else:
            if data < self.data: #If data is smaller than self.data
                if self.left is None:
                    self.left = Shop(data) #If there is nothing in the left side, the left side gets the data of the tree
                else:
                    self.left.Insert(data) #else the data in the left side gets inserted into the tree
            elif data > self.data:
                if self.right is None
                    self.right = Shop(data) #If there is nothing in the right side, the right side gets the data of the tree
                else:
                    self.right.Insert(data) #else the data in the right side gets inserted into the tree

    def Printtree(self): #Function to print data in the tree
        if self.left: #If data is found on the left side
            self.left.Printtree() #The data of the left side is printed
        print(self.data), #Prints the data found
        if self.right: #If data is found on the right side
            self.right.Printtree() #The data of the right side is printed

root = Shop('Groceries') #Gives the main node of the Shop tree the value 'Groceries'
root.Insert('\t |__Pasteries') #Give all the nodes below their values
root.Insert('\t |__Fruits')
root.Insert('\t |__Vegetables')
root.Insert('\t |__Meats')
#Give all the subheadings below their values
root.Insert('\t\t |__Bread')
root.Insert('\t\t |__Cake')
root.Insert('\t\t |__Cheese Griller')
root.Insert('\t\t |__Meat Pie')

root.Insert('\t\t |__Apples')
root.Insert('\t\t |__Oranges')
root.Insert('\t\t |__Mangoes')
root.Insert('\t\t |__Pawpaw')

root.Insert('\t\t |__Broccoli')
root.Insert('\t\t |__Carrots')
root.Insert('\t\t |__Potatoes')
root.Insert('\t\t |__Okra')

root.Insert('\t\t |__Beef')
root.Insert('\t\t |__Chicken')
root.Insert('\t\t |__Turkey')
root.Insert('\t\t |__Pork')

root.Printtree() #Prints the Shop Tree
```

Input for Building Tree:

```
class Building:  #Create the class for the 2nd Tree
    def __init__(self, data):
        self.left = None #Create a variable for the left node of the tree
        self.right = None #Create a variable for the right node of the tree
        self.data = data #Variable for passing and entering data

    def Insert(self, data): #Function to insert data into the tree
        if self.data == None: #If there is no value in data
            self.data = data  #Self.data gets a value from data
        else:
            if data < self.data: #If data is smaller than self.data
                if self.left is None:
                    self.left = Building(data) #If there is nothing in the left side, the left side gets the data of the tree
                else:
                    self.left.Insert(data) #else the data in the left side gets inserted into the tree
            elif data > self.data:
                if self.right is None
                    self.right = Building(data) #If there is nothing in the right side, the right side gets the data of the tree
                else:
                    self.right.Insert(data) #else the data in the right side gets inserted into the tree

     def Printtree(self): #Function to print data in the tree
        if self.left: #If data is found on the left side
            self.left.Printtree() #The data of the left side is printed
        print(self.data), #Prints the data found
        if self.right: #If data is found on the right side
            self.right.Printtree() #The data of the right side is printed

root = Building('Building 1') #Gives the main node of the Building tree the value 'Building 1'
root.Insert('\t |__South') #Give all the nodes below their values
root.Insert('\t |__East')
root.Insert('\t |__West')
root.Insert('\t |__North')
#Give all the subheadings below their values
root.Insert('\t\t |__South - Shelf 1')
root.Insert('\t\t |__South - Shelf 2')
root.Insert('\t\t |__South - Shelf 3')
root.Insert('\t\t |__South - Shelf 4')

root.Insert('\t\t |__East - Shelf 1')
root.Insert('\t\t |__East - Shelf 2')
root.Insert('\t\t |__East - Shelf 3')
root.Insert('\t\t |__East - Shelf 4')

root.Insert('\t\t |__West - Shelf 1')
root.Insert('\t\t |__West - Shelf 2')
root.Insert('\t\t |__West - Shelf 3')
root.Insert('\t\t |__West - Shelf 4')

root.Insert('\t\t |__North - Shelf 1')
root.Insert('\t\t |__North - Shelf 2')
root.Insert('\t\t |__North - Shelf 3')
root.Insert('\t\t |__North - Shelf 4')

root.Printtree() #Prints the Building Tree
```

Input for Combined Tree:

```
class Combined:  #Create the class for the 3rd Tree
    def __init__(self, data):
        self.left = None #Create a variable for the left node of the tree
        self.right = None #Create a variable for the right node of the tree
        self.data = data #Variable for passing and entering data

    def Insert(self, data): #Function to insert data into the tree
        if self.data == None: #If there is no value in data
            self.data = data  #Self.data gets a value from data
        else:
            if data < self.data: #If data is smaller than self.data
                if self.left is None:
                    self.left = Combined(data) #If there is nothing in the left side, the left side gets the data of the tree
                else:
                    self.left.Insert(data) #else the data in the left side gets inserted into the tree
            elif data > self.data:
                if self.right is None
                    self.right = Combined(data) #If there is nothing in the right side, the right side gets the data of the tree
                else:
                    self.right.Insert(data) #else the data in the right side gets inserted into the tree

    def Printtree(self): #Function to print data in the tree
        if self.left: #If data is found on the left side
            self.left.Printtree() #The data of the left side is printed
        print(self.data), #Prints the data found
        if self.right: #If data is found on the right side
            self.right.Printtree() #The data of the right side is printed

root = Combined('Groceries (Building 1)')#Gives the main node of the Combined tree the value 'Groceries(Building 1)'
root.Insert('\t |__Pasteries(South)')#Give all the nodes below their values
root.Insert('\t |__Fruits(East)')
root.Insert('\t |__Vegetables(West)')
root.Insert('\t |__Meats(North)')
#Give all the subheadings below their values
root.Insert('\t\t |__Bread(South - Shelf 1)')
root.Insert('\t\t |__Cake(South - Shelf 2)')
root.Insert('\t\t |__Cheese Griller(South - Shelf 3)')
root.Insert('\t\t |__Meat Pie(South - Shelf 4)')

root.Insert('\t\t |__Apples(East - Shelf 1)')
root.Insert('\t\t |__Oranges(East - Shelf 2)')
root.Insert('\t\t |__Mangoes(East - Shelf 3)')
root.Insert('\t\t |__Pawpaw(East - Shelf 4)')

root.Insert('\t\t |__Broccoli(West - Shelf 1)')
root.Insert('\t\t |__Carrots(West - Shelf 2)')
root.Insert('\t\t |__Potatoes(West - Shelf 3)')
root.Insert('\t\t |__Okra(West - Shelf 4)')

root.Insert('\t\t |__Beef(North - Shelf 1')
root.Insert('\t\t |__Chicken(North - Shelf 2')
root.Insert('\t\t |__Turkey(North - Shelf 3')
root.Insert('\t\t |__Pork(North - Shelf 4')

root.Printtree()#Prints the Combined Tree
```

Output:

```
                    |__Apples
                    |__Beef
                    |__Bread
                    |__Broccoli
                    |__Cake
                    |__Carrots
                    |__Cheese Griller
                    |__Chicken
                    |__Mangoes
                    |__Meat Pie
                    |__Okra
                    |__Oranges
                    |__Pawpaw
                    |__Pork
                    |__Potatoes
                    |__Turkey
          |__Fruits
          |__Meats
          |__Pasteries
          |__Vegetables
Groceries
                    |__East - Shelf 1
                    |__East - Shelf 2
                    |__East - Shelf 3
                    |__East - Shelf 4
                    |__North - Shelf 1
                    |__North - Shelf 2
                    |__North - Shelf 3
                    |__North - Shelf 4
                    |__South - Shelf 1
                    |__South - Shelf 2
                    |__South - Shelf 3
                    |__South - Shelf 4
                    |__West - Shelf 1
                    |__West - Shelf 2
                    |__West - Shelf 3
                    |__West - Shelf 4
          |__East
          |__North
          |__South
          |__West
Building 1
                    |__Apples(East - Shelf 1)
                    |__Beef(North - Shelf 1
                    |__Bread(South - Shelf 1)
                    |__Broccoli(West - Shelf 1)
                    |__Cake(South - Shelf 2)
                    |__Carrots(West - Shelf 2)
                    |__Cheese Griller(South - Shelf 3)
                    |__Chicken(North - Shelf 2
                    |__Mangoes(East - Shelf 3)
                    |__Meat Pie(South - Shelf 4)
                    |__Okra(West - Shelf 4)
                    |__Oranges(East - Shelf 2)
                    |__Pawpaw(East - Shelf 4)
                    |__Pork(North - Shelf 4
                    |__Potatoes(West - Shelf 3)
                    |__Turkey(North - Shelf 3
          |__Fruits(East)
          |__Meats(North)
          |__Pasteries(South)
          |__Vegetables(West)
Groceries (Building 1)
```

3.2

Input:

```
class GroceriesStack: #Create a class for the stack
    def __init__(self):
        self.Stack = [] #Data will be entered in self.Stack

    def GroceryCategory(self, DATA): #Create Function GroceryCategory that will add values to the stack
        if DATA not in self.Stack: #If the current value is not present in the stack
            self.Stack.append(DATA) #Add the value to the stack
            return True #Value was added
        else:
            return False #Value was not added

    def Atop(self): #Create a function that will return the value that is above the function when it is called
        return self.Stack[-1] #Return the value above

Groceries = GroceriesStack() #Create the class variable
Groceries.GroceryCategory('Pastries') #Insert values into the stack
Groceries.Atop() #Call the atop function
Groceries.GroceryCategory('Meats')
print(Groceries.Atop()) #Will print the value above, 'Meats'
Groceries.GroceryCategory('Fruits')
Groceries.GroceryCategory('Vegetables')
print(Groceries.Atop()) #Will print the value above, 'Vegetables'
```

Output:

```
Meats
Vegetables
```

3.3

Input:

```
HashFoods = {'Bread':14.10, 'Cake':22.20, 'Meat Pie':12.90, 'Cheese Griller':23.60, 'Apples':2.10,
             'Oranges':1.20, 'Mangoes':2.60, 'PawPaw':2.90, 'Broccoli':10.20, 'Carrots':4.30, 'Potatoes':99.90,
            'Okra':60.50, 'Beef':100.20, 'Chicken':210.11, 'Turkey':102.11, 'Pork':80.35} #Create the Hashtable

def RetrievePrice(): #Create the RetrievePrice function to receive the price of a specific item searched by the user
    Product = input('Type which product price to see:') #The user must enter a product to search for
    for X in HashFoods: #X searches for the product in the Hashtable
        if Product == X: #If the product is found in the Hashtable
            print(HashFoods[X]) #Prints the price of the searched product

def DeleteProduct(): #Create the DeleteProduct function to delete an item from the Hashtable
    Product = input('Type which product to remove:') #Asks the user which product they want to remove from the Hashtable
    del HashFoods[Product] #Deletes the product from the Hashtable
    print('The Product', Product, 'is deleted:', HashFoods) #Prints the hashtable without the deleted product

def SetPrice(): #Create the SetPrice function to update the price of a specific item searched by the user
    Product = input('Type a product to be updated:') #Asks the user for a product name
    HashFoods[Product] = int(input("Enter the new Price")) #Asks for the new price of the product
    print('The new price of', Product, 'is:', HashFoods[Product]) #Prints the name of the product and its updated value
    print('')
    print('The Updated list:', HashFoods) #Prints the Hashtable with its updated values

def Selection(): #Enables the user to make a selection between the three functions
    Choice = int(input("Please make a choice: 1:Retrieve a product price, 2: Delete a product, 3:Update a product price "))
    if Choice == 1:
        RetrievePrice() #Calls the RetrievePrice funtion
    if Choice == 2:
        DeleteProduct() #Calls the DeleteProduct funtion
    if Choice == 3:
        SetPrice() #Calls the SetPrice funtion

Selection() #Calls the selection function to ask the user which function they want to use
```

Output:

1.

```
Please make a choice: 1:Retrieve a product price, 2: Delete a product, 3:Update a product price 1
Type which product price to see:Bread
14.1
```

2.

```
Please make a choice: 1:Retrieve a product price, 2: Delete a product, 3:Update a product price 2
Type which product to remove:Bread
The Product Bread is deleted: {'Cake': 22.2, 'Meat Pie': 12.9, 'Cheese Griller': 23.6, 'Apples': 2.1, 'Oranges': 1.2, 'Mangoe
s': 2.6, 'PawPaw': 2.9, 'Broccoli': 10.2, 'Carrots': 4.3, 'Potatoes': 99.9, 'Okra': 60.5, 'Beef': 100.2, 'Chicken': 210.11, 'Tu
rkey': 102.11, 'Pork': 80.35}
```

3.

```
Please make a choice: 1:Retrieve a product price, 2: Delete a product, 3:Update a product price 3
Type a product to be updated:Bread
Enter the new Price15
The new price of Bread is: 15

The Updated list: {'Bread': 15, 'Cake': 22.2, 'Meat Pie': 12.9, 'Cheese Griller': 23.6, 'Apples': 2.1, 'Oranges': 1.2, 'Mangoe
s': 2.6, 'PawPaw': 2.9, 'Broccoli': 10.2, 'Carrots': 4.3, 'Potatoes': 99.9, 'Okra': 60.5, 'Beef': 100.2, 'Chicken': 210.11, 'Tu
rkey': 102.11, 'Pork': 80.35}
```

Bibliography:

1. N/A, 2021, Python Binary Tree, Tutorialspoint.com
   Available at: https://www.tutorialspoint.com/python_data_structure/python_binary_tree.htm
   [Date Accessed: 25 November 2021]
2. Kindson The Genius, 2021, Introduction to trees (Binary Tree) in Python, Youtube.com
   https://www.youtube.com/watch?v=fUkrQD9nw0Y
   [Date Accessed: 25 November 2021]
3. N/A, 2021, Python File readlines() Method, w3schools.com
   https://www.w3schools.com/python/ref_file_readlines.asp
   [Date Accessed: 23 November 2021]
4. N/A, 2021, Create linked list from a given array, geeksforgeeks.org
   https://www.geeksforgeeks.org/create-linked-list-from-a-given-array/
   [Date Accessed: 26 November 2021]
5. N/A, 2021, Reverse an array in python, askpython.com
   https://www.askpython.com/python/array/reverse-an-array-in-python
   [Date Accessed: 26 November 2021]
6. N/A, 2021, Python 3 File readlines() method, Tutorialspoint.com
   https://www.tutorialspoint.com/python3/file_readlines.htm
   [Date Accessed: 24 November 2021]
7. N/A, 2021, Introduction of B-tree, geeksforgeeks.org
   https://www.geeksforgeeks.org/introduction-of-b-tree-2/
   [Date Accessed: 25 November 2021]
8. N/A, 2021, Python linked list: Create a singly linked list, append some items and iterate through the list
   https://www.w3resource.com/python-exercises/data-structures-and-algorithms/python-linked-list-exercise-1.php
   [Date Accessed: 25 November 2021]
9. Sanatan, M, 2019, Stacks and Queues in Python, stackabuse.com
   https://stackabuse.com/stacks-and-queues-in-python/
   [Date Accessed: 26 November 2021]
10. N/A, 2021, Stack in Python, geeksforgeeks.org
    https://www.geeksforgeeks.org/stack-in-python/
    [Date Accessed: 26 November 2021]