Started on	Wednesday, 14 December 2022, 10:25 AM
State	Finished
Completed on	Wednesday, 14 December 2022, 11:34 AM
Time taken	1 hour 9 mins
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 2.00 out of 2.00

Write a Python Program to find whether a string is a palindrome or not using recursion

For example:

Input	Result
civic	String is a palindrome

Answer: (penalty regime: 0 %)

Reset answer

```
def palindrome(word):
    if(string==string[::-1]):
        return"String is a palindrome"
    else:
        return"String is not a palindrome"
string=str(input())
print(palindrome(string))
```

	Input	Expected	Got	
✔	madam	String is a palindrome	String is a palindrome	✔
✔	civic	String is a palindrome	String is a palindrome	✔
✔	church	String is not a palindrome	String is not a palindrome	✔
✔	mom	String is a palindrome	String is a palindrome	✔
✔	lal	String is a palindrome	String is a palindrome	✔

Passed all tests! ✔

Question author's solution (Python3):

```
def is_palindrome(word):
    if len(word) <= 1:
        return True
    else:
        return word[0] == word[-1] and is_palindrome(word[1:-1])


str=input()
if(is_palindrome(str)):
```

```
    print("String is a palindrome")
else:
    print("String is not a palindrome")

```

Correct

Marks for this submission: 2.00/2.00.

Question **2**

Correct

Mark 2.00 out of 2.00

Write Python program to solve the tower of Hanoi problem for n disks

Answer: (penalty regime: 0 %)

```
def towerOfHanoi (n,source,destination,auxillary) :
    if n==1:
        print("Move disk 1 from source",source,"to destination",destination)
        return
    towerOfHanoi(n-1,source,auxillary,destination)
    print("Move disk",n,"from source",source,"to destination",destination)
    towerOfHanoi(n-1,auxillary,destination,source)
n=int(input())
towerOfHanoi(n,"A","C","B")
```

	Input	Expected	Got	
✔	3	Move disk 1 from source A to destination C Move disk 2 from source A to destination B Move disk 1 from source C to destination B Move disk 3 from source A to destination C Move disk 1 from source B to destination A Move disk 2 from source B to destination C Move disk 1 from source A to destination C	Move disk 1 from source A to destination C Move disk 2 from source A to destination B Move disk 1 from source C to destination B Move disk 3 from source A to destination C Move disk 1 from source B to destination A Move disk 2 from source B to destination C Move disk 1 from source A to destination C	✔
✔	4	Move disk 1 from source A to destination B Move disk 2 from source A to destination C Move disk 1 from source B to destination C Move disk 3 from source A to destination B Move disk 1 from source C to destination A Move disk 2 from source C to destination B Move disk 1 from source A to destination B Move disk 4 from source A to destination C Move disk 1 from source B to destination C Move disk 2 from source B to destination A Move disk 1 from source C to destination A Move disk 3 from source B to destination C Move disk 1 from source A to destination B Move disk 2 from source A to destination C Move disk 1 from source B to destination C	Move disk 1 from source A to destination B Move disk 2 from source A to destination C Move disk 1 from source B to destination C Move disk 3 from source A to destination B Move disk 1 from source C to destination A Move disk 2 from source C to destination B Move disk 1 from source A to destination B Move disk 4 from source A to destination C Move disk 1 from source B to destination C Move disk 2 from source B to destination A Move disk 1 from source C to destination A Move disk 3 from source B to destination C Move disk 1 from source A to destination B Move disk 2 from source A to destination C Move disk 1 from source B to destination C	✔

Passed all tests! ✔

Question author's solution (Python3):

```
def TowerofHanoi(n , source, destination, auxiliary):
    if n==1:
        print ("Move disk 1 from source",source,"to destination",destination)
```

```
        print ("Move disk 1 from source",source,"to destination",destination)
        return
    TowerofHanoi(n-1, source, auxiliary, destination)
    print ("Move disk",n,"from source",source,"to destination",destination)
    TowerofHanoi(n-1, auxiliary, destination, source)

n = int(input())
TowerofHanoi(n,'A','C','B')

```

Correct

Marks for this submission: 2.00/2.00.

Question **3**

Correct

Mark 2.00 out of 2.00

Write a program to search a word in a list of n words using binary search.

For example:

Test	Input	Result

index_of_element = binary_search(my_list, 0, len(my_list)-1, element_to_search)
if index_of_element != -1:
  print("Element searched is found at the index ", str(index_of_element), "of given list")
else:
  print("Element searched is not found in the given list!")
	bat
	3
	cat
	bat
	sort
		The given list is
		['cat', 'bat', 'sort']
		['bat', 'cat', 'sort']
		Element searched is found at the index 0 of given list

Answer: (penalty regime: 0 %)

Reset answer

```
def create_list():
    no_of_elements = int(input())
    array = []
    for i in range (0,no_of_elements,1):
        element=input()
        array.append(element)
    return (array)

def binary_search(L, start, end, item):
    if end >= start:
        middle = (start+end)//2
        if L[middle]==item :
            return middle
        elif L[middle] > item :
            return binary_search (L,start,middle-1,item)
        else:
            return binary_search(L,middle+1,end,item)
    else:
        return -1

my_list = []
element to search = input()
```

	Test	Input	Expected	Got	

✔	index_of_element = binary_search(my_list, 0, len(my_list)-1, element_to_search)
	if index_of_element != -1:
	  print("Element searched is found at the index ", str(index_of_element), "of given list")
	else:
	  print("Element searched is not found in the given list!")
		bat
		3
		cat
		bat
		sort
			The given list is
			['cat', 'bat', 'sort']
			['bat', 'cat', 'sort']
			Element searched is found at the index 0 of given list
				The given list is
				['cat', 'bat', 'sort']
				['bat', 'cat', 'sort']
				Element searched is found at the index 0 of given list
					✔

Passed all tests! ✔

Question author's solution (Python3):

```
def binary_search(L, start, end, item):
```

```
    if end >= start:
        middle = (start + end) // 2
        if L[middle] == item:
            return middle
        elif L[middle] > item:
            return binary_search(L, start, middle - 1, item)
        else:
            return binary_search(L, middle + 1, end, item)
    else:
        return -1

my_list = [ ]
element_to_search = input()
n=int(input())
for i in range(n):
    x=input()
    my_list.append(x)
print("The given list is")
print(my_list)
my_list=sorted(my_list)
print(my_list)
```

Correct

Marks for this submission: 2.00/2.00.

Question **4**

Correct

Mark 2.00 out of 2.00

Write a Python Program to sort n real numbers using merge sort algorithm with recursion

For example:

Test	Input	Result
alist = create_list() mergeSort(alist) print(alist)	5 23.45 98.67 11.77 84.22 66.48	Enter the size of listSplitting [23.45, 98.67, 11.77, 84.22, 66.48] Splitting [23.45, 98.67] Splitting [23.45] Merging [23.45] Splitting [98.67] Merging [98.67] Merging [23.45, 98.67] Splitting [11.77, 84.22, 66.48] Splitting [11.77] Merging [11.77] Splitting [84.22, 66.48] Splitting [84.22] Merging [84.22] Splitting [66.48] Merging [66.48] Merging [66.48, 84.22] Merging [11.77, 66.48, 84.22] Merging [11.77, 23.45, 66.48, 84.22, 98.67] [11.77, 23.45, 66.48, 84.22, 98.67]

Answer: (penalty regime: 0 %)

Reset answer

```
def create_list():
    no_of_elements = int(input('Enter the size of list'))
    array =[]
    for i in range(0,no_of_elements,1):
        element=input()
        array.append(float(element))
    return (array)

def mergeSort(alist):
    print("Splitting ",alist)
    if len(alist)>1:
        mid = len(alist)//2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]
        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=0
        j=0
        k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                alist[k]=lefthalf[i]
```

	Test	Input	Expected	Got	
✔	alist = create_list() mergeSort(alist) print(alist)	5 23.45 98.67 11.77 84.22 66.48	Enter the size of listSplitting [23.45, 98.67, 11.77, 84.22, 66.48] Splitting [23.45, 98.67] Splitting [23.45] Merging [23.45] Splitting [98.67] Merging [98.67] Merging [23.45, 98.67] Splitting [11.77, 84.22, 66.48] Splitting [11.77] Merging [11.77] Splitting [84.22, 66.48] Splitting [84.22] Merging [84.22] Splitting [66.48] Merging [66.48] Merging [66.48, 84.22] Merging [11.77, 66.48, 84.22] Merging [11.77, 23.45, 66.48, 84.22, 98.67] [11.77, 23.45, 66.48, 84.22, 98.67]	Enter the size of listSplitting [23.45, 98.67, 11.77, 84.22, 66.48] Splitting [23.45, 98.67] Splitting [23.45] Merging [23.45] Splitting [98.67] Merging [98.67] Merging [23.45, 98.67] Splitting [11.77, 84.22, 66.48] Splitting [11.77] Merging [11.77] Splitting [84.22, 66.48] Splitting [84.22] Merging [84.22] Splitting [66.48] Merging [66.48] Merging [66.48, 84.22] Merging [11.77, 66.48, 84.22] Merging [11.77, 23.45, 66.48, 84.22, 98.67] [11.77, 23.45, 66.48, 84.22, 98.67]	✔

Passed all tests! ✔

Question author's solution (Python3):

```
def create_list():
    no_of_elements = int(input('Enter the size of list'))
    array =[]
    for i in range(0,no_of_elements,1):
        element=input()
        array.append(float(element))
    return (array)
def mergeSort(alist):
    print("Splitting ",alist)
    if len(alist)>1:
        mid = len(alist)//2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]
        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=0
        j=0
        k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                alist[k]=lefthalf[i]
                i=i+1
```

Correct

Marks for this submission: 2.00/2.00.

Question **5**

Correct

Mark 2.00 out of 2.00

Given a string, the task is to check and accept the given string if contains all vowels i.e. 'a', 'e','i'.'o', 'u' . Use Recursion

Examples :

Input : Welcome

Output : Not Accepted

Input : education

Output : Accepted

For example:

Input	Result
Welcome	Not Accepted

Answer: (penalty regime: 0 %)

```
def ispresent(str,l):
    if len(l)<=0:
        return True
    else:
        if(l[0] in str):
            return (True and ispresent(str,l[1:]))
l=['a','e','i','o','u']
str=input()
if(ispresent(str,l)):
    print("Accepted")
else:
    print("Not Accepted")

```

	Input	Expected	Got	
✔	Welcome	Not Accepted	Not Accepted	✔
✔	education	Accepted	Accepted	✔

Passed all tests! ✔

Question author's solution (Python3):

```
def ispresent(str,l):
    if len(l)<=0:
        return True
    else:
        if(l[0] in str):
            return (True and ispresent(str,l[1:]))
l=['a','e','i','o','u']
```

```
str=input()
if(ispresent(str,l)):
    print("Accepted")
else:
    print("Not Accepted")

```

Correct

Marks for this submission: 2.00/2.00.