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))
8
9
```

	Input	Expected	Got	
~	✓ madam String is a palindrome String is a part of the part o		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! ✓

```
1 v
def is_palindrome(word):
    if len(word) <= 1:
        return True
    else:
        return word[0] == word[-1] and is_palindrome(word[1:-1])
6
7
8    str=input()
9 v</pre>
```

```
print("String is a palindrome")

print("String is not a palindrome")

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

Marks for this submission: 2.00/2.00.

1.

```
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) :
2
        if n==1:
3
            print("Move disk 1 from source", source, "to destination", destination)
4
            return
        towerOfHanoi(n-1,source,auxillary,destination)
        print("Move disk",n,"from source",source,"to destination",destination)
6
7
        towerOfHanoi(n-1,auxillary,destination,source)
8
    n=int(input())
    towerOfHanoi(n,"A","C","B")
9
10
```

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

Passed all tests! 🗸

```
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')
```

Marks for this submission: 2.00/2.00.

1.

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

Answer: (penalty regime: 0 %)

Reset answer

```
1 • def create_list():
 2
        no_of_elements = int(input())
 3
        array = []
 4 ·
        for i in range (0,no_of_elements,1):
 5
            element=input()
 6
            array.append(element)
 7
        return (array)
 8
 9 •
    def binary_search(L, start, end, item):
10
        if end >= start:
            middle = (start+end)//2
11
            if L[middle]==item :
12 ,
13
                return middle
            elif L[middle] > item :
14
15
                return binary_search (L,start,middle-1,item)
16
            else:
17
                return binary_search(L,middle+1,end,item)
        else:
18
19
            return -1
20
    my_list = []
22 | element to search = input()
```

	Test	Input	Expected	Got	
•	<pre>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!")</pre>	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! ✓

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

```
if end >= start:
3
            middle = (start + end) // 2
            if L[middle] == item:
4
5
                return middle
            elif L[middle] > item:
6
7
                return binary_search(L, start, middle - 1, item)
8
            else:
9
                return binary_search(L, middle + 1, end, item)
10 •
        else:
11
            return -1
12
13
    my_list = []
14
    element_to_search = input()
15
    n=int(input())
    for i in range(n):
16
17
        x=input()
18
        my_list.append(x)
    print("The given list is")
19
20
   print(my_list)
21
   my_list=sorted(my_list)
22 | print(my_list)
```

Marks for this submission: 2.00/2.00.

1.

```
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()	5	Enter the size of listSplitting [23.45, 98.67, 11.77, 84.22, 66.48]
mergeSort(alist)	23.45	Splitting [23.45, 98.67]
print(alist)	98.67	Splitting [23.45]
	11.77	Merging [23.45]
	84.22	Splitting [98.67]
	66.48	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

```
1 def create_list():
        no_of_elements = int(input('Enter the size of list'))
 2
 3
        array =[]
 4
        for i in range(0,no_of_elements,1):
 5
             element=input()
 6
             array.append(float(element))
 7
        return (array)
 8
    def mergeSort(alist):
 9
        print("Splitting ",alist)
10
11
        if len(alist)>1:
12
             mid = len(alist)//2
13
             lefthalf = alist[:mid]
             righthalf = alist[mid:]
14
15
             mergeSort(lefthalf)
             mergeSort(righthalf)
16
17
             i=0
18
             j=<mark>0</mark>
19
             k=0
             while i < len(lefthalf) and j < len(righthalf):</pre>
20
                 if lefthalf[i] < righthalf[j]:</pre>
21 •
22
                     alist[k]=lefthalf[i]
```

	Test	Input	Expected	Got	
~	alist =	5	Enter the size of listSplitting	Enter the size of listSplitting	~
	<pre>create_list()</pre>	23.45	[23.45, 98.67, 11.77, 84.22, 66.48]	[23.45, 98.67, 11.77, 84.22, 66.48]	
	mergeSort(alist)	98.67	Splitting [23.45, 98.67]	Splitting [23.45, 98.67]	
	print(alist)	11.77	Splitting [23.45]	Splitting [23.45]	
		84.22	Merging [23.45]	Merging [23.45]	
		66.48	Splitting [98.67]	Splitting [98.67]	
			Merging [98.67]	Merging [98.67]	
			Merging [23.45, 98.67]	Merging [23.45, 98.67]	
			Splitting [11.77, 84.22, 66.48]	Splitting [11.77, 84.22, 66.48]	
			Splitting [11.77]	Splitting [11.77]	
			Merging [11.77]	Merging [11.77]	
			Splitting [84.22, 66.48]	Splitting [84.22, 66.48]	
			Splitting [84.22]	Splitting [84.22]	
			Merging [84.22]	Merging [84.22]	
			Splitting [66.48]	Splitting [66.48]	
			Merging [66.48]	Merging [66.48]	
			Merging [66.48, 84.22]	Merging [66.48, 84.22]	
			Merging [11.77, 66.48, 84.22]	Merging [11.77, 66.48, 84.22]	
			Merging [11.77, 23.45, 66.48, 84.22,	Merging [11.77, 23.45, 66.48, 84.22,	
			98.67]	98.67]	
			[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):

```
1 def create_list():
 2
        no_of_elements = int(input('Enter the size of list'))
 3
        array =[]
 4 ·
        for i in range(0,no_of_elements,1):
             element=input()
 5
 6
             array.append(float(element))
        return (array)
 7
    def mergeSort(alist):
 8
 9
        print("Splitting ",alist)
10
        if len(alist)>1:
             mid = len(alist)//2
11
             lefthalf = alist[:mid]
12
13
             righthalf = alist[mid:]
14
             mergeSort(lefthalf)
15
             mergeSort(righthalf)
16
             i=0
17
             j=<mark>0</mark>
             k=0
18
             while i < len(lefthalf) and j < len(righthalf):</pre>
19
20 •
                 if lefthalf[i] < righthalf[j]:</pre>
21
                     alist[k]=lefthalf[i]
22
                     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 %)

```
1 ▼ def ispresent(str,1):
 2 🔻
        if len(1)<=0:</pre>
 3
            return True
 4
        else:
 5 •
            if(l[0] in str):
                 return (True and ispresent(str,l[1:]))
 6
    l=['a','e','i','o','u']
 7
 8
    str=input()
    if(ispresent(str,1)):
 9 •
10
        print("Accepted")
11 v else:
12
        print("Not Accepted")
13
```

Input		Input Expected		
~	Welcome	Not Accepted	Not Accepted	~
~	education	Accepted	Accepted	~

Passed all tests! 🗸

```
8
9 v
if(ispresent(str,1)):
    print("Accepted")
11 v
else:
    print("Not Accepted")
13
```

Marks for this submission: 2.00/2.00.