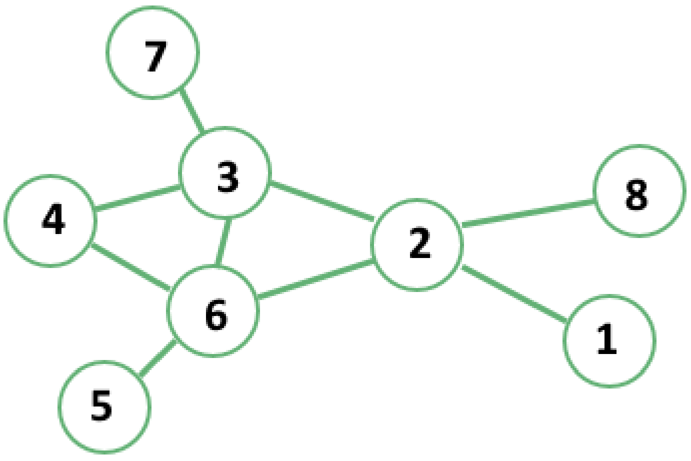
End Term Mock [1 Hour]

Question 1-5

A graph G can be defined as an ordered set G(V, E) where V represents the set of vertices and E represents the set of edges.

For Example



Undirected Graph

In the image:

- Vertices: V= {1, 2, 3, 4, 5, 6, 7, 8}
- Edges : E = {(1, 2), (2, 8), (2, 3), (2, 6), (3, 6), (3, 4), (4, 6), (5, 6), (3, 7)},
- In the undirected graph, edge (u, v) and (v, u) represent the same edge. Therefore, we can consider any one of them.

This graph can be represented in Dictionary (`G = {vertex : [list of adjacent vertices]}`) like this:

```
G = {1: [2],
     2: [1, 3, 6, 8],
     3: [2, 4, 6, 7],
     4: [3, 6],
     5: [6],
     6: [2, 3, 4, 5],
     7: [3],
     8: [2]}
```

In the below code, `graph` is a dictionary of the undirected graph and `vertex` is a vertex name that belongs to vertices of `graph`. Assume that `graph` doesn't have any self loop and more than one edge between any pair(`u, v`) of vertex.

```python
def f(graph, vertex):
    return len(graph[vertex])

def g(graph):
    return len(graph)

def h(graph):
    T = 0
    for v in graph:
        T += len(graph[v])
    return T // 2

def Regular_graph(graph):
    # fill code
    return Reg

def complete_graph(graph):
    # fill code
    return comp
```

Question 1

Degree of vertex `v`: It is the number of adjacent vertices to a vertex `v`. Function `f(graph, vertex)` returns degree of `vertex` in `graph` .

(a) True, it returns the degree of `vertex`

(b) False, it does not return the degree of `vertex`

Answer

(a)

Solution

Each key contains a list of the adjacent vertex in the dictionary `graph`, So the length of the list represents the degree of a vertex. Hence option (a) is correct.

Question 2

For above graph G, what function `g(G)` will return ? [NAT]

Answer

8

Solution

Function `g(G)` is returning the length of `G` which is 8. So the correct answer is 8.

Question 3

Function `h(graph)` returns number of edges in `graph` .

(a) True, it returns the number of edges

(b) False, it does not return the number of edges

Answer

(a)

Solution

One edge incident on two vertexes or for any edge (u,v) we can say that v is the adjacent of u and u is the adjacent of v. So the function `h` returns the number of edges by dividing the total number of adjacent by 2. Hence Option (a) is correct.

Question 4

A regular graph is a graph in which all vertex has an equal degree(or an equal number of the adjacent vertices).

Which of the following code is correct to fill in the function `Regular_graph(graph)` to check `graph` is regular or not?

(a)

```
Reg = True
deg = 0
for v in graph:
    if deg == 0:
        deg = f(graph, v)
    else:
        if deg != f(graph, v):
            Reg = False
        else:
            break
```

(b)

```
Reg = True
deg = 0
for v in graph:
    if deg == 0:
        deg = f(graph, v)
    else:
        if deg != f(graph, v):
            Reg = False
        break
```

(c)

```python
Reg = True
deg = 0
for v in graph:
    if deg == 0:
        deg = f(graph, v)
    else:
        if deg != f(graph, v):
            Reg = False
            break
```

(d)

```python
Reg = True
deg = 0
for v in graph:
    if deg == 0:
        deg = f(graph, v)
    else:
        if deg == f(graph, v):
            Reg = False
            break
```

Answer

(c)

Solution

If all vertex has an equal number of adjacent then it is called a Regular graph. So it can be checked by comparing the length of each key's value. So option (c) is the correct code to check graph is regular or not.

Question 5

A complete graph is a graph in which every vertex connects to the remaining vertex by a direct edge. So if graph has `n` vertices then degree of each vertex should be `n-1` for complete graph.

Which of the following code is correct to fill in function `complete_graph(graph)` to check `graph` is a complete graph or not?

(a)

```python
comp = True
deg = 0
for v in graph:
    if deg == 0:
        deg = f(graph, v)
        if deg != len(graph)-1:
            comp = True
    else:
        if deg != f(graph, v):
            comp = False
            break
```

(b)

```python
comp = True
deg = 0
for v in graph:
    if deg == 0:
        deg = f(graph, v)
        if deg != len(graph)-1:
            comp = False
            break
    else:
        if deg != f(graph, v):
            comp = False
            break
```

(c)

```python
comp = True
deg = 0
for v in graph:
    if deg == 0:
        deg = f(graph, v)
        if deg != len(graph):
            comp = False
    else:
        if deg == f(graph, v):
            comp = False
            break
```

(d)

```python
comp = True
deg = 0
for v in graph:
    if deg == 0:
        deg = f(graph, v)
        if deg == len(graph)-1:
            comp = False
    else:
        if deg != f(graph, v):
            comp = False
            break
```

Answer

(b)

Solution

If all vertex have `total vertex in graph - 1` adjacent then it called Complete graph. So it can be checked by comparing the length of each key's value. So option (b) is the correct code to check graph is complete or not.

Question 6-9

Suppose `L` is a list of integers that is already initialized.

```python
def P(L):
    a = 0
    for i in range(len(L)):
        if L[i] not in L[:i]:
            a += 1
    return a

def Q(L):
    b = 0
    for i in range(len(L)):
        if L[i] in L[:i]:
            if L[i] not in L[i + 1:]:
                b += 1
    return b

def R(L):
    x = 0
    c = 0
    for i in range(len(L)):
        if L[i] > c:
            x = c
            c = L[i]
        else:
            if L[i] > x:
                x = L[i]
    return x

def S(L):
    d = 0
    for i in range(len(L)):
        if L[i] not in L[:i]:
            y = 1
        else:
            if L[i] != L[i - 1]:
                y = 1
            else:
                y += 1
                if y > d:
                    z = L[i]
                    d = y
    return z
```

Question 6

What `P(L)` returns after execution of the above code?

(a) Total number of elements in the list `L`.

(b) Total number of elements that are repeated in the list `L` more than one time.

(c) Total number of distinct elements in the list `L.`

(d) Largest element of the list `L`

Answer

(c)

Question 7

What `Q(L)` returns after execution of the above code?

(a) Total number of elements in the list `L`.

(b) Total number of elements that are repeated in the list `L` more than one time.

(c) Total number of distinct elements in the list `L.`

(d) Second largest element of the list `L`

Answer

(b)

Question 8

What `R(L)` returns after execution of the above code?

(a) Smallest element of the list `L`

(b) Largest element of the list `L`

(c) Second largest element of the list `L`

(d) Second smallest element of the list `L`

Answer

(c)

Question 9

What `S(L)` returns after execution of the above code?

(a) The first element which is consecutive repeated more than one time in list `L`

(b) The first element which is consecutively repeated maximum number of times in the list `L`

(c) The last element which is consecutively repeated maximum number of times in the list `L`

(d) the Last element which is consecutively repeated more than one time in the list `L`

Answer

(b)

Solution (Question 6 - 9)

Take a random list with duplicate or consecutively repeated elements. Execute each function and observe the output and flow of execution line by line.

Question 10

Which of the following options can be used to produce first `n` Fibonacci numbers starting from `0`. It is a Multiple Select Question (MSQ).

```
0, 1, 1, 2, 3, 5, 8, 13,..., upto n terms
```

(a)

```python
def fib(n):
    a, b = 0, 1
    for i in range(0, n):
        a, b = b, a + b
    return a
for i in range(n):
    print(fib(i))
```

(b)

```python
def fib(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    return fib(n - 2) + fib(n - 1)
for i in range(n):
    print(fib(i))
```

(c)

```python
def fib(n):
    a, b = 0, 1
    for i in range(n):
        yield a
        a, b = b, a + b
for i in fib(n):
    print(i)
```

(d) None of these

Answer

(a), (b) and (c)

Solution

Option (a), (b) and (c) returns first `n` Fibonacci numbers starting from `0`. Execute each code and observe the output and flow of execution.

Question 11-13

Consider the following function `f` that accepts a positive integer as an argument `n`.

```python
def f(n):
    if n > 1:
        if (n ** 0.5) == int(n ** 0.5):
            return 1 + f(int(n ** 0.5))
        elif n % 2 != 0:
            return 1 + f(3 * n + 1)
        else:
            return 1 + f(int(n / 2))
    elif n < 1:
        return 1 + f(n ** 2)
    else:
        return 0
```

Question 11

`if (n ** 0.5) == int(n ** 0.5):` conditional statement checks `n` is a square of any positive integer or not.

(a) True

(b) False

Answer

(a)

Solution

`if (n ** 0.5) == int(n ** 0.5):` if n is not a perfect square then this conditional statement will return `False` otherwise it will return `True`. Hence option (a) is correct.

Question 12

What is the value returned by `f(10000)`? [NAT]

Answer

7

Solution

```
f(10000)
1 + f(100)
1 + 1 + f(10)
1 + 1 + 1 + f(5)
1 + 1 + 1 + 1 + f(16)
1 + 1 + 1 + 1 + 1 + f(4)
1 + 1 + 1 + 1 + 1 + 1 + f(2)
1 + 1 + 1 + 1 + 1 + 1 + 1 + f(1)
1 + 1 + 1 + 1 + 1 + 1 + 1 + 0
output = 7
```

Question 13

What is the value returned by `f(-8)`? [NAT]

Answer

5

Solution

```
f(-8)
1 + f(64)
1 + 1 + f(8)
1 + 1 + 1 + f(4)
1 + 1 + 1 + 1 + f(2)
1 + 1 + 1 + 1 + 1 + f(1)
1 + 1 + 1 + 1 + 1 + 0
output = 5
```

Question 14

`L` is a non-empty list of distinct integers that has already been defined. Which of the following recursive functions return a sorted list in ascending order? It is a Multiple Select Question (MSQ).

(a)

```python
def sort(L):
    if L != []:
        max_index = L.index(max(L))
        L[max_index], L[-1] = L[-1], L[max_index]
        return L[-1:] + sort(L[:-1])
    else:
        return L
```

(b)

```python
def sort(L):
    if L != []:
        max_index = L.index(max(L))
        L[max_index], L[-1] = L[-1], L[max_index]
        return sort(L[:-1]) + L[-1:]
    else:
        return L
```

(c)

```python
def sort(L):
    if L != []:
        min_index = L.index(min(L))
        L[min_index], L[0] = L[0], L[min_index]
        return  L[:1] + sort(L[1:])
    else:
        return L
```

(d)

```python
def sort(L):
    if L != []:
        min_index = L.index(min(L))
        L[min_index],L[0] = L[0], L[min_index]
        return  sort(L[1:]) + L[:1]
    else:
        return L
```

Answer

(b) and (c)

Solution

Option (b) and (c) are the correct recursive code for sort the elements of the list in ascending order.

In option (b), in each call of function, the max element and last position element will be swap and `sort` the function will call again on the remaining list element except for the last indexed element.

In option (c), in each call of function, the min element and first position element will be swap and `sort` the function will call again on the remaining list element except for the first element of the list.

Question 15

What function `foo(10)` will return?

```python
def foo(k):
    try:
        k = p
        if k == p:
```

```
 5              raise ValueError
 6          else:
 7              raise NameError
 8      except ValueError:
 9          return 'ValueError'
10      except NameError:
11          return 'NameError'
12      except:
13          return 'SyntaxError'
14      return 'NoError'
```

(a) `ValueError`

(b) `NameError`

(c) `SyntaxError`

(d) `NoError`

Answer

(b)

Solution

In line 3, variable `p` is not defined in the function. Therefore `NameError` will occur. Option (b) is correct

Question 16

```
class test:
    def __init__(self):
        self.variable = 'Old'
        self.Change(self.variable)
    def Change(self, var):
        var = 'New'
obj = test()
print(obj.variable)
```

What will be output for the following code?

(a) Error because function change can’t be called in the `__init__` function

(b) `New` is printed

(c) `Old` is printed

(d) Nothing is printed

Answer

(c)

Solution

In constructor, we are passing the value of `self.variable` in `change` method which is `old` and in method `change` we are updating the local variable `var` with `new` so `self.variable` will remain unchanged. Hence option (c) is correct.

Question 17

```python
class Test:
    print('This is class message 1')
    def __init__(self):
        print('This is object message')
    print('This is class message 2')
a = Test()
b = Test()
```

What will be output for the above code after execution?

(a)

```
This is object message
This is object message
```

(b)

```
This is class message 1
This is class message 2
This is object message
This is object message
```

(c)

```
This is class message 1
This is object message
This is class message 2
This is object message
```

(d)

```
This is class message 1
This is object message
This is class message 2
This is class message 1
This is object message
This is class message 2
```

Answer

(b)

Solution

The compiler executes the code in a sequential manner. So first of all when the class is initialized then print statement(line 2)(`This is class message 1` and (line 5) `This is class message 2`) will be printed after that when the object will be created then the constructor will be called and will be printed two times for two objects. Hence option (b) is correct.

Question 18

```
f = open("hello.txt", "w")
f.write("Hello World how are you today")
f.close()

f = open("hello.txt", "r")
data = f.readlines()
for line in data:
    words = line.split()
    print(words)
f.close()
```

What will be output for the above code?

(a) Runtime Error

(b) `Hello World how are you today`

(c) `['Hello', 'World', 'how', 'are', 'you', 'today']`

(d) `Hello`

Answer

(c)

Solution

File have only one line `Hello World how are you today`. So after reading the file, `line.split()` will return a list of word (`['Hello', 'World', 'how', 'are', 'you', 'today']`) after split the string by space.

Question 19

```
l = []
for i in range(10):
    if(i % 2 == 0):
        l.append(i)
    else:
        l.append("Odd")
```

Which of the following codes are equivalent to the above code snippet? It is a Multiple Select Question (MSQ).

(a)

```
l = [i if i % 2 == 0: else: "Odd" for i in range(10)]
```

(b)

```
l = [i if i % 2 == 0 else "Odd" for i in range(10)]
```

(c)

```
l = ["Odd" if i % 2 != 0 else i for i in range(10)]
```

(d)

```
l = [l.append(i) if i % 2 == 0 else "Odd" for i in range(10)]
```

Answer

(b) and (c)

Solution

Option (b) and (c) are the correct equivalent list comprehension code for given code.

Question 20

```
a = [10, 20, 30, 40, 50]
b = [1, 2, 3, 4, 5, 6, 7]
L = list(map(lambda n, m: m * n, a, b))
```

What will be the value of `L` after executing the above code snippet?

(a)

```
[10, 40, 90, 160, 250, 6, 7]
```

(b)

```
[10, 40, 90, 160, 250]
```

(c)

```
[10, 40, 90, 160, 250, 300, 350]
```

(d) `IndexError`

Answer

(b)

Solution

In this code `map` the function will repeatedly take value from the list `a` and `b` and pass to the lambda function. if any time any list element is completed then the mapping process will be stopped.

```
output = [10*1, 20*2, 30*3, 40*4, 50*5] = [10, 40, 90, 160, 250]
```

Hence option (b) is correct.