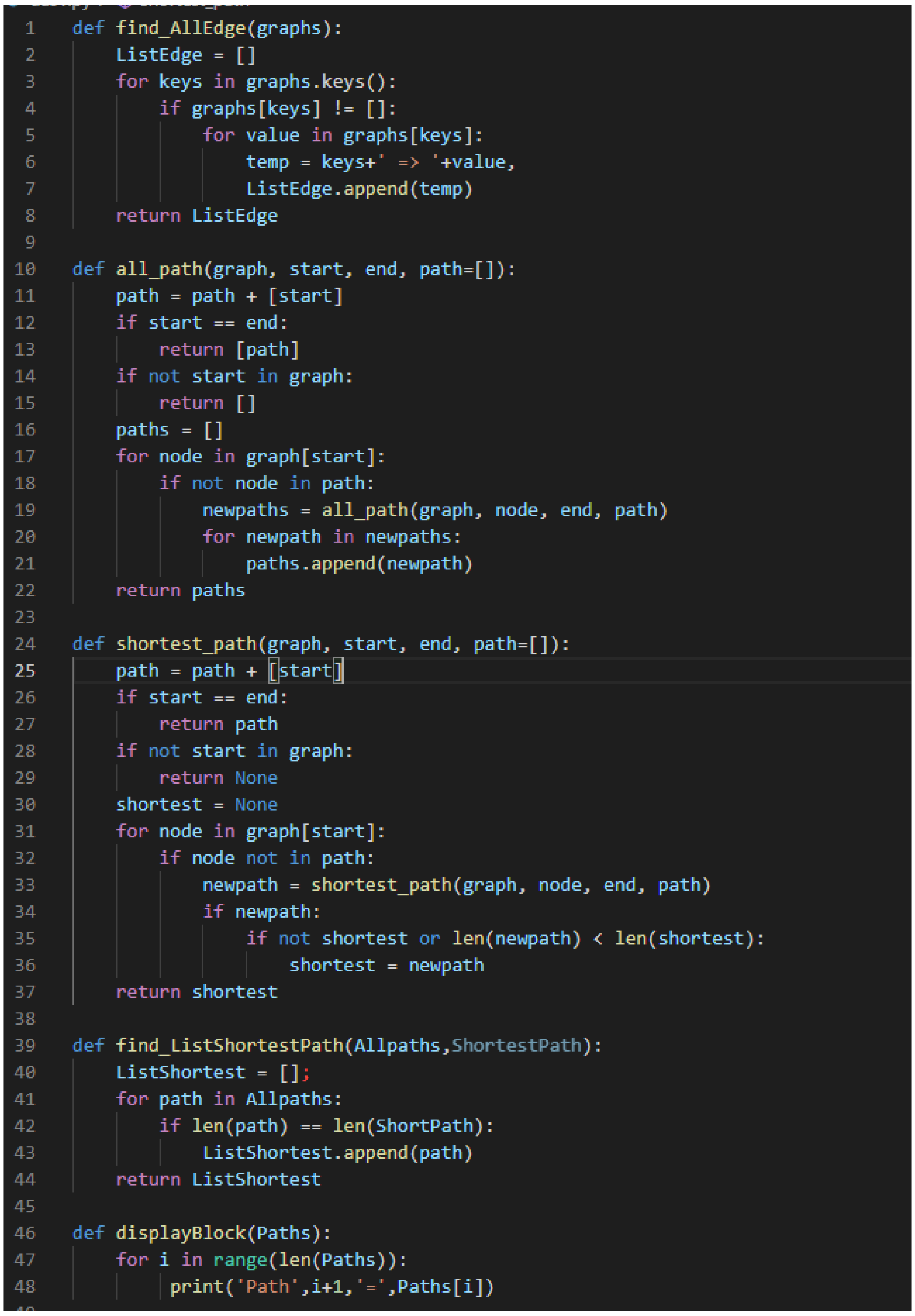
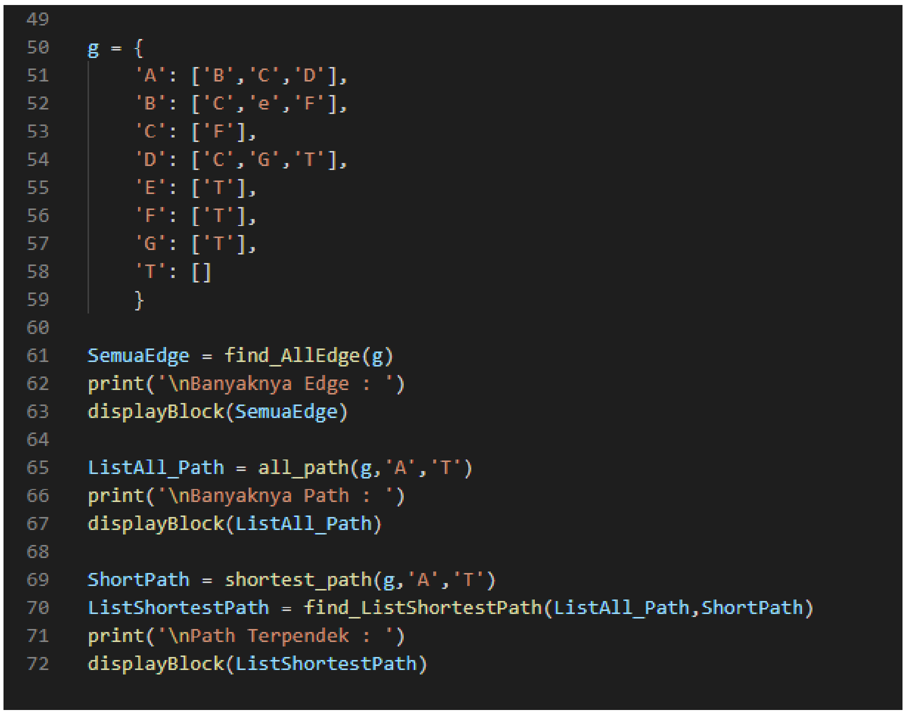
Nama : I Made Landiva

NIM : 2201010591

Soal no 1

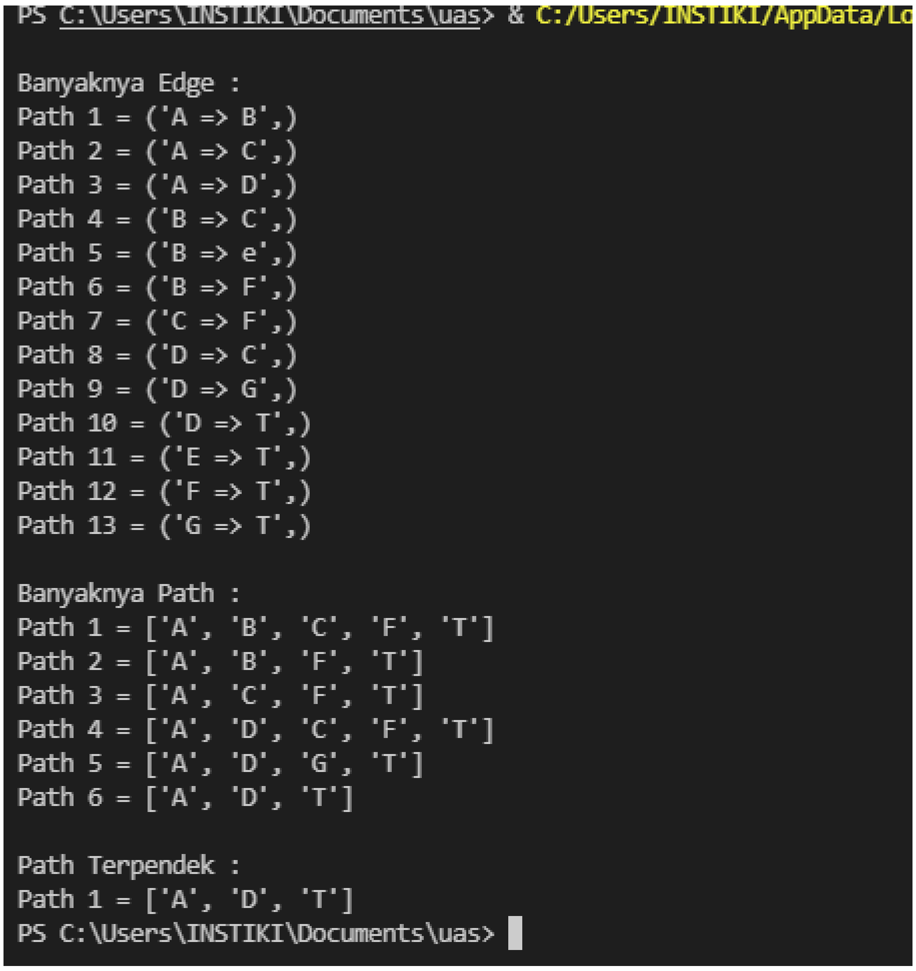


```python
def find_AllEdge(graphs):
    ListEdge = []
    for keys in graphs.keys():
        if graphs[keys] != []:
            for value in graphs[keys]:
                temp = keys+' => '+value,
                ListEdge.append(temp)
    return ListEdge

def all_path(graph, start, end, path=[]):
    path = path + [start]
    if start == end:
        return [path]
    if not start in graph:
        return []
    paths = []
    for node in graph[start]:
        if not node in path:
            newpaths = all_path(graph, node, end, path)
            for newpath in newpaths:
                paths.append(newpath)
    return paths

def shortest_path(graph, start, end, path=[]):
    path = path + [start]
    if start == end:
        return path
    if not start in graph:
        return None
    shortest = None
    for node in graph[start]:
        if node not in path:
            newpath = shortest_path(graph, node, end, path)
            if newpath:
                if not shortest or len(newpath) < len(shortest):
                    shortest = newpath
    return shortest

def find_ListShortestPath(Allpaths,ShortestPath):
    ListShortest = [];
    for path in Allpaths:
        if len(path) == len(ShortPath):
            ListShortest.append(path)
    return ListShortest

def displayBlock(Paths):
    for i in range(len(Paths)):
        print('Path',i+1,'=',Paths[i])
```



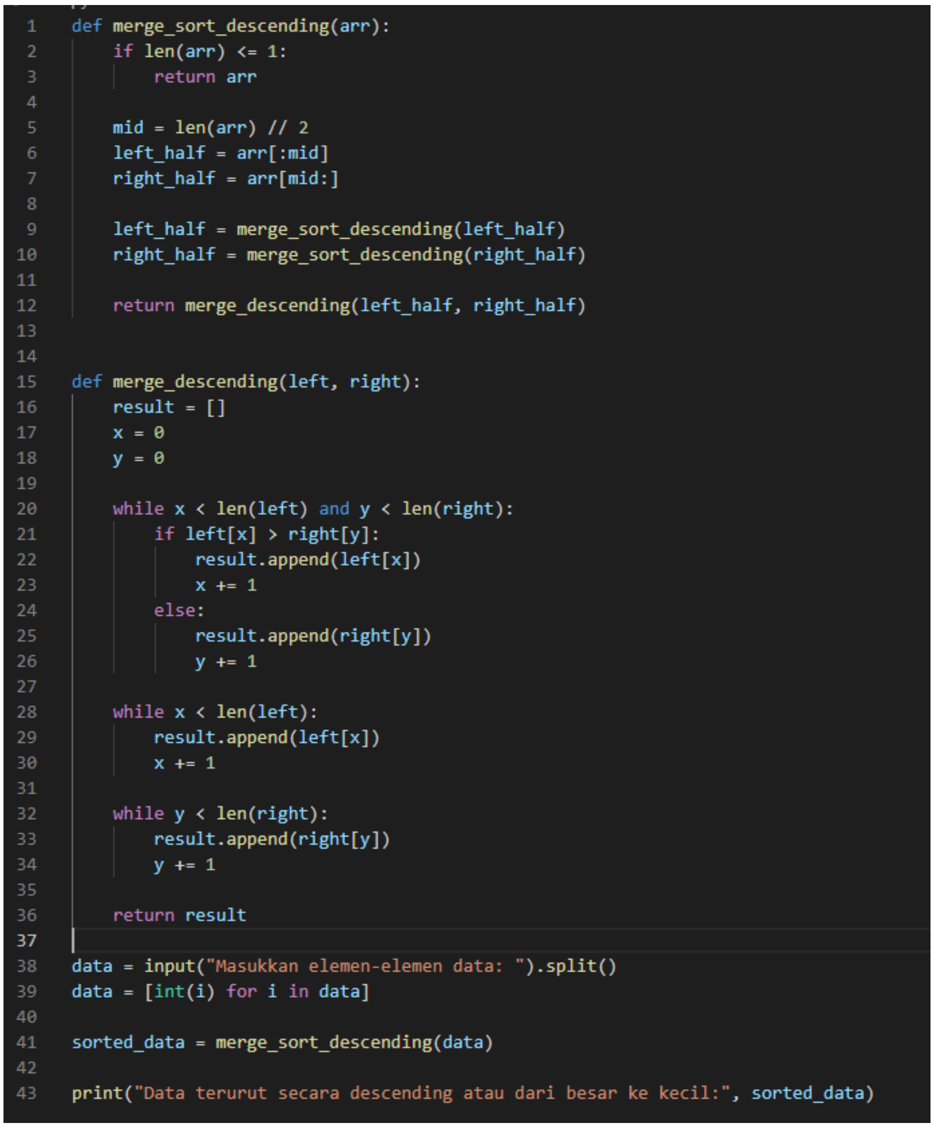
```python

g = {
    'A': ['B','C','D'],
    'B': ['C','e','F'],
    'C': ['F'],
    'D': ['C','G','T'],
    'E': ['T'],
    'F': ['T'],
    'G': ['T'],
    'T': []
    }

SemuaEdge = find_AllEdge(g)
print('\nBanyaknya Edge : ')
displayBlock(SemuaEdge)

ListAll_Path = all_path(g,'A','T')
print('\nBanyaknya Path : ')
displayBlock(ListAll_Path)

ShortPath = shortest_path(g,'A','T')
ListShortestPath = find_ListShortestPath(ListAll_Path,ShortPath)
print('\nPath Terpendek : ')
displayBlock(ListShortestPath)
```

Hasil run:



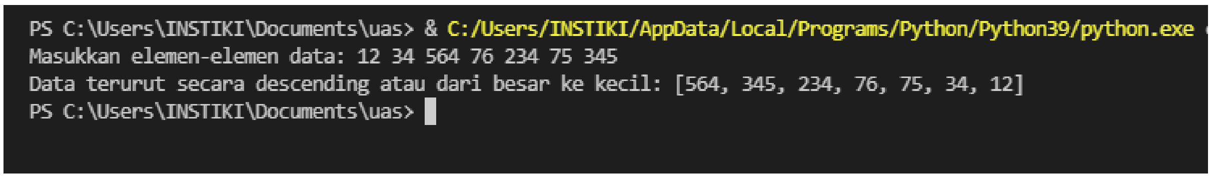
```
PS C:\Users\INSTIKI\Documents\uas> & C:/Users/INSTIKI/AppData/Lo

Banyaknya Edge :
Path 1 = ('A => B',)
Path 2 = ('A => C',)
Path 3 = ('A => D',)
Path 4 = ('B => C',)
Path 5 = ('B => e',)
Path 6 = ('B => F',)
Path 7 = ('C => F',)
Path 8 = ('D => C',)
Path 9 = ('D => G',)
Path 10 = ('D => T',)
Path 11 = ('E => T',)
Path 12 = ('F => T',)
Path 13 = ('G => T',)

Banyaknya Path :
Path 1 = ['A', 'B', 'C', 'F', 'T']
Path 2 = ['A', 'B', 'F', 'T']
Path 3 = ['A', 'C', 'F', 'T']
Path 4 = ['A', 'D', 'C', 'F', 'T']
Path 5 = ['A', 'D', 'G', 'T']
Path 6 = ['A', 'D', 'T']

Path Terpendek :
Path 1 = ['A', 'D', 'T']
PS C:\Users\INSTIKI\Documents\uas>
```

Soal no 2



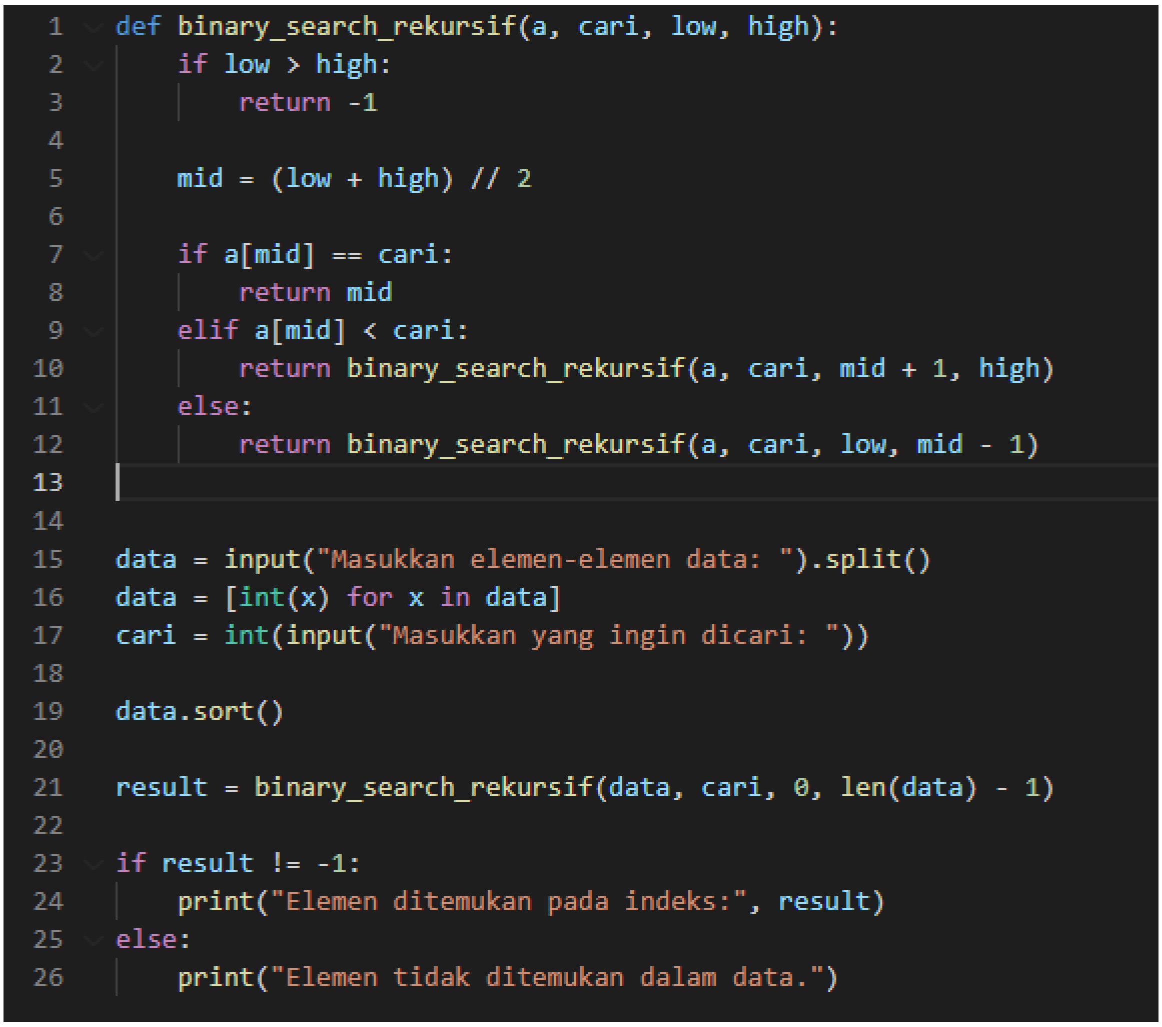
```python
def merge_sort_descending(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    left_half = merge_sort_descending(left_half)
    right_half = merge_sort_descending(right_half)

    return merge_descending(left_half, right_half)


def merge_descending(left, right):
    result = []
    x = 0
    y = 0

    while x < len(left) and y < len(right):
        if left[x] > right[y]:
            result.append(left[x])
            x += 1
        else:
            result.append(right[y])
            y += 1

    while x < len(left):
        result.append(left[x])
        x += 1

    while y < len(right):
        result.append(right[y])
        y += 1

    return result

data = input("Masukkan elemen-elemen data: ").split()
data = [int(i) for i in data]

sorted_data = merge_sort_descending(data)

print("Data terurut secara descending atau dari besar ke kecil:", sorted_data)
```

Hasil run:



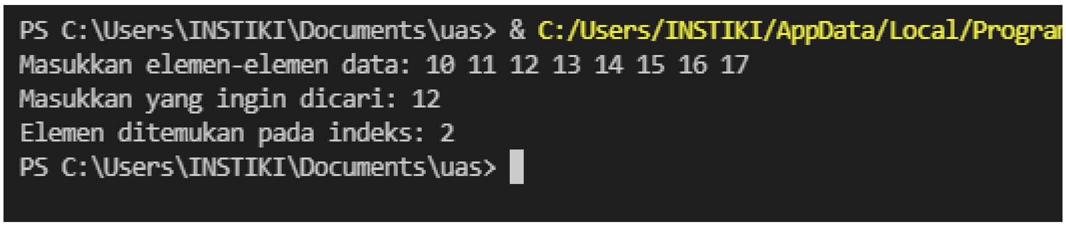
```
PS C:\Users\INSTIKI\Documents\uas> & C:/Users/INSTIKI/AppData/Local/Programs/Python/Python39/python.exe
Masukkan elemen-elemen data: 12 34 564 76 234 75 345
Data terurut secara descending atau dari besar ke kecil: [564, 345, 234, 76, 75, 34, 12]
PS C:\Users\INSTIKI\Documents\uas>
```

Soal no 3



```python
def binary_search_rekursif(a, cari, low, high):
    if low > high:
        return -1

    mid = (low + high) // 2

    if a[mid] == cari:
        return mid
    elif a[mid] < cari:
        return binary_search_rekursif(a, cari, mid + 1, high)
    else:
        return binary_search_rekursif(a, cari, low, mid - 1)


data = input("Masukkan elemen-elemen data: ").split()
data = [int(x) for x in data]
cari = int(input("Masukkan yang ingin dicari: "))

data.sort()

result = binary_search_rekursif(data, cari, 0, len(data) - 1)

if result != -1:
    print("Elemen ditemukan pada indeks:", result)
else:
    print("Elemen tidak ditemukan dalam data.")
```

Hasil run:



```
PS C:\Users\INSTIKI\Documents\uas> & C:/Users/INSTIKI/AppData/Local/Prograr
Masukkan elemen-elemen data: 10 11 12 13 14 15 16 17
Masukkan yang ingin dicari: 12
Elemen ditemukan pada indeks: 2
PS C:\Users\INSTIKI\Documents\uas>
```