Nama : Nicolas Pascal Bagaskara

NIM/NPM : 23.01.5014

Ruang : 4.3.2

No Kursi Ujian : 25

**Soal Tentang Materi List :**

1. Apakah list dan apakah fungsinya jelaskan
2. Untuk mengimplementasikan fungsi length, yaitu dengan menghitung panjang node dari sebuah list. Alternatif lainnya adalah dengan menyimpan jumlah simpul dalam list sebagai data tambahan yang diletakkan di head dari list. “Ubahlah class UList berikut untuk dapat menyertakan informasi tersebut dan tulis ulang fungsi size”

Kode Program Sebelum di Ubah :

class UList:

def \_\_init\_\_(self):

self.head = None

def isEmpty(self):

return self.head == None

def add(self,item):

temp = Node(item)

temp.setNext(self.head)

self.head = temp

def size(self):

current = self.head

count = 0

while current != None:

count = count + 1

current = current.getNext()

return count

**Jawaban :**

1. List adalah struktur data yang digunakan untuk menyimpan sejumlah elemen dalam urutan tertentu. Dalam pemrograman, list biasanya berbentuk kumpulan data atau elemen yang bisa diakses, ditambah, dihapus, atau diubah sesuai kebutuhan.

**Fungsi Utama List dalam Pemrograman:**

* Pengelompokan Data: Menyimpan dan mengelompokkan data yang berhubungan dalam satu variabel, seperti daftar nama atau angka.
* Memudahkan Pengolahan Data: List membuat proses pemrosesan data lebih mudah karena mendukung perulangan (looping) untuk operasi batch pada semua elemen.
* Efisiensi Penyimpanan Data: Dengan list, kita bisa menyimpan sejumlah besar data dalam satu tempat tanpa harus membuat variabel terpisah untuk setiap elemen.

1. Kode program :

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

def setNext(self, next\_node):

self.next = next\_node

def getNext(self):

return self.next

class UList:

def \_\_init\_\_(self):

self.head = Node(None) # Menggunakan head sebagai node dummy dengan count

self.head.count = 0 # Inisialisasi jumlah node dalam daftar

def isEmpty(self):

return self.head.getNext() is None # Memeriksa apakah daftar kosong

def add(self, item):

temp = Node(item)

temp.setNext(self.head.getNext())

self.head.setNext(temp)

self.head.count += 1 # Update jumlah elemen setiap kali ada elemen baru yang ditambahkan

def size(self):

return self.head.count # Mengembalikan nilai count dari head langsung

Stack adalah struktur data LIFO (Last In, First Out), di mana elemen terakhir yang ditambahkan akan menjadi elemen pertama yang dikeluarkan. Stack digunakan dalam berbagai penerapan. Berikut beberapa penerapan berbeda dari stack

class Stack:

def \_\_init\_\_(self):

self.items = []

def is\_empty(self):

return len(self.items) == 0

def push(self, item):

self.items.append(item)

def pop(self):

if not self.is\_empty():

return self.items.pop()

def peek(self):

if not self.is\_empty():

return self.items[-1]

def size(self):

return len(self.items)

def decimal\_to\_binary(decimal\_number):

stack = Stack()

# Handle the special case for 0

if decimal\_number == 0:

return "0"

# Convert decimal to binary by dividing by 2 and pushing the remainder to the stack

while decimal\_number > 0:

remainder = decimal\_number % 2

stack.push(remainder)

decimal\_number = decimal\_number // 2

# Pop all elements from the stack to get the binary representation

binary\_number = ""

while not stack.is\_empty():

binary\_number += str(stack.pop())

return binary\_number

# Contoh penggunaan

decimal\_number = int(input("Masukkan bilangan desimal: "))

print("Hasil konversi ke biner:", decimal\_to\_binary(decimal\_number))

Queue adalah salah satu struktur data linear yang bekerja dengan prinsip FIFO (First In First Out), artinya elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang keluar. Struktur ini mirip dengan antrian pada kehidupan sehari-hari, di mana orang pertama yang datang adalah orang pertama yang dilayani.

class Queue:

def \_\_init\_\_(self):

self.items = []

def is\_empty(self):

return len(self.items) == 0

def enqueue(self, item):

self.items.insert(0, item)

def dequeue(self):

if not self.is\_empty():

return self.items.pop()

def size(self):

return len(self.items)

def is\_palindrome(word):

# Menghilangkan spasi dan mengubah ke huruf kecil untuk membuat pengecekan tidak sensitif terhadap kapital

word = word.replace(" ", "").lower()

# Membuat dua queue, satu untuk memeriksa dari depan dan satu dari belakang

queue1 = Queue()

queue2 = Queue()

# Tambahkan setiap karakter ke kedua queue

for char in word:

queue1.enqueue(char) # Karakter masuk dari belakang

queue2.enqueue(char) # Karakter masuk dari belakang

# Bandingkan elemen dari depan queue1 dan belakang queue2

for \_ in range(len(word) // 2):

if queue1.dequeue() != queue2.items.pop(0): # Mengambil dari depan queue2

return False # Jika tidak sama, bukan palindrom

return True # Jika sama, maka palindrom

# Contoh penggunaan

word = input("Masukkan kata atau kalimat: ")

if is\_palindrome(word):

print("Kata atau kalimat tersebut adalah palindrom.")

else:

print("Kata atau kalimat tersebut bukan palindrom.")