**Stack**

* Tumpukan, pengoperasian yang ditambahkan diakhir tumpukan dan begitupun pengambilannya paling terakhir. Stack menggunakan konsep LIFO (Last in first out)
* Operasi pada stack :

stack () = inisialisasi stack

Push (data) = menambahkan data kedalam list

Pop() = mengambil data dari paling terakhir yg dimasukkan

Peek() = mengecek data dari paling terakhir yg dimasukkan

isEmpty() = mengecek apakah suatu stack dalam keadaan kosong apa tidak

Size() = mengetahui panjang/jumlah data suatu stack

* Stack harus dibuat fungsinya terlebih dahulu :

def stack():

st = []

return (st)

def push(st,data):

st.append(data)

def pop(st):

data=st.pop()

return(data)

def peek(st):

return(st[len(st)-1])

def isEmpty(st):

return (st==[])

def size(st):

return(len(st))

* Pembalikkan Kata

|  |
| --- |
| def balik(teks):  a = stack()  hasil = ''  for i in range(len(teks)):  push(a,teks[i])  for i in range(size(a)):  hasil = hasil + pop(a)  return hasil  teks = input('kata : ')  print (balik(teks)) |

* Delimiter matching (memeriksa keseimbangan jumlah kurung) : Perhatikan tiap tanda, operand, dan operator

Pop kurung buka :

1. Stack kosong
2. Sejenis
3. Cocok = True

|  |
| --- |
| def stack():  st = []  return (st)  def push(st,data):  st.append(data)  def pop(st):  data=st.pop()  return(data)  def peek(st):  return(st[len(st)-1])  def isEmpty(st):  return (st==[])  def size(st):  return(len(st))  #Rumus  def cekTanda(tanda):  st = stack()  isi = stack()  for i in tanda:  if i == '(' :  push(st,i)  else :  if i == ')' :  if i == ')' and not (isEmpty(st)):  pop(st)  elif i == ')' :  push(isi,i)  if len(st) > 0 and len(isi) == 0 :  print ('Kelebihan ( = ',len(st) )  elif len(isi) > 0 and len(st) == 0 :  print ('Kelebihan ) = ',len(isi) )  elif len(st) and len(isi) > 0 :  print ('kelebihan ( = ',len(st),'------','kelebihan ) = ',len(isi))  else :  print ('DATA BENAR')  nilai = input('aaa : ')  cekTanda(nilai) |

* Konversi desimal ke oktal

|  |
| --- |
| Def oktal(angka):  st = stack()    while angka > 0:  sisa = angka%8  push(st,sisa)  angka = angka // 8  oktal = ' '  while not isEmpty(st):  oktal = oktal+str(pop(st))  return oktal  a = int(input('angka : '))  print (oktal(a)) |

* Infix, Prefix, dan Postfix dalam Matematika

Disini menggunakan inputan huruf

Algoritma dari Infix ke Postfix :

- Ketemu operand masukkan ke output

-- Ketemu operator

- Pop semua operator yg levelnya(presedence) lebih tinggi atau sama dengan (digunakan saat operator yang dibaca adalah + dan - selama tidak kosong (empty) )

Kalau Sama dengan(=) adalah bagi 2 jenis operator yg level tinggi dan rendah

- Kurung buka -> push

- Kurung tutup -> semua data distack di pop sampai ada kurung buka

Contoh :

A\*B+C postfixnya BC+A\*

A+B\*C postfixnya BC\*A+

Stacknya :

A = output

+

B

\* = pop semua yg sama dengan atau yg lebih tinggi

Push \*

C

|  |
| --- |
| def stack():  s=[]  return (s)  def push (s,data):  s.append(data)  def pop(s):  data = s.pop()  return (data)  def peek (s):  return(s[len(s)-1])  def isEmpty(s):  return (s==[])  def size (s):  return(len(s))  def infixPostfix(mathStr):  highPre='\*/'  lowPre='-+'  operator=highPre+lowPre  postStr=''  st=stack()  for i in mathStr:  if i in lowPre:  if isEmpty(st):  push(st,i)  else:  temp=peek(st)  #print('peek=',peek(st))  while (temp in operator) and not(isEmpty(st)) :  postStr=postStr+pop(st)  #print('while')  push(st,i)  elif i in highPre:  if isEmpty(st):  push(st,i)  else:  while not(isEmpty(st)) and (peek(st)in highPre):  postStr=postStr\*pop(st)  push(st,i)  elif i =='(':  push(st,i)  elif i==')':  while not(isEmpty(st)) and peek(st)!='(':  postStr=postStr+pop(st)  pop(st)  else:  postStr=postStr+i  #print(postStr)  #print('stack=',st)  while not(isEmpty(st)):  postStr=postStr+pop(st)    print(postStr)    infixPostfix('(A+B)\*(C+D)') |