1. STACK

STACK adalah penyimpanan dari item – item yang berurutan tergantung pada bagaimana item tersebut ditambahkan atau di hapus. Stack memiliki dua ujung yang sama . ujung ini dinamakan “top” dan ujung berlawanan di namakan “base” dengan ini stack mengikuti konsep LIFO (last in first out).
Contoh module stack :

stack.py

```
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))
```

Operasi dalam stack :

stack()	Membuat stack baru atau sebuah penyimpanan dan tidak menggunakan paramater
push(stack,item)	Menambahkan suatu item baru di dalam stack dan memerlukan parameter stack dan itemnya
pop(stack)	Menghapus suatu item teratas di dalam stack dan memerlukan parameter stack
peek(stack)	Mengembalikan nilai item teratas dari sebuah stack dan memerlukan paramater dan stack tidak berubah
isEmpty(stack)	Memeriksa isi di dalam stack dan outputnya bisa jadi type data bolean (True and False)
size(stack)	Mengembalikan jumlah item di dalam stack dan memerlukan sebuah paramater.

1.1. Contoh pembuatan pembalikan sebuah kata contoh kata yang di inputkan adalah “AKU” jadi outputnya akan menjadi “UKA”. Jadi algoritmanya kita membuat sebuah perulangan untuk memasukan sebuah huruf A,K,U di dalam stack(), lalu setiap huruf di pop() dan dimasukkan kedalam variable string .
Contoh :

```
From stack import *
def rev(s):
    my = stack()
    for i in range(len(s)):
        push(my,s[i])
    kata = ''
    for i in range (size(my)):
        kata = kata + my.pop()
    return kata
i = input("Masukkan kata = ")
print (rev(i))
```

1.2. Memeriksa keseimbangan jumlah tanda kurung, Algoritmanya di mulai dari stack kosong, jadi ketika kita menginputkan semisal “2 x (2+3)” operasi matematikannya benar outpunya data benar , jika kita menginputkan “(2x(2+3)” opersi matematikanya salah karena kurang “)” berjumlah 1 , jadi jika ada suatu simbol “(“ maka push ke stack dan jika ada simbol “)” maka pop stacknya . ketika semua simbol sudah di proses , stack akan kosong.
Contoh :

```
From stack import *
def cek(tanda):
  sAwal = stack()
  sAkhir = stack()
  for i in tanda:
      if i == "(":
          push(sAwal,i)
      elif i == ")":
          if i == ")" and isEmpty(sAwal)==False:
              pop(sAwal)
          else:
              push(sAkhir,i)
  if (size(sAwal) > 0 and size(sAkhir) == 0) :
      print("kelebihan ( = ",size(sAwal))
  elif (size(sAkhir) > 0 and size(sAwal) == 0):
      print("kelebihan ) = ",size(sAkhir))
  elif (size(sAwal) > 0 and size(sAkhir) > 0):
      print("kelebihan ( = %s dan kelebihan ) = %s" %
(size(sAwal),size(sAkhir)))
  else:
      print("Data Benar")


simbol = input("Masukan operasi = ")
cek(simbol)
```

1.3. Konversi bilangan Desimal ke binner menggunakan algoritma modulus 2 dan bagi 2
Contoh code :

```
from stack import *
    def con(num):
        s = stack()
        while num > 0:
                a = num % 2
                push(s,a)
                num = num // 2
        string = ""
        while not isEmpty(s):
        string = string + str(pop(s))
                return string
    n = int(input("masukkan angka = "))
print(con(n))
```

1.4. Konversi bilangan Desimal ke Oktal menggunakan algoritma modulus 8 dan bagi 8
Contoh code:

```
from stack import *
def con(num):
        s = stack()
        while num > 0:
                a = num % 8
                push(s,a)
                num = num // 8
        string = ""
        while not isEmpty(s):
                string = string + str(pop(s))
        return string
 n = int(input("masukkan angka = "))
print(con(n))
```

1.5. Konversi bilangan Desimal ke Oktal menggunakan algoritma modulus 8 dan bagi 8
Contoh :

```
from stack import *
def con(num):
s = stack()
while num > 0:
        a = num % 16
        if (a == 10):
                push(s,"A")
        elif (a == 11):
                push(s,"B")
        elif (a == 12):
                push(s,"C")
        elif (a == 13):
                push(s,"D")
        elif (a == 14):
                push(s,"E")
        elif (a == 15):
                push(s,"F")
        else:
                push(s,a)
        num = num // 16
string = ""
while not isEmpty(s):
        string = string + str(pop(s))

return string
n = int(input("masukkan angka = "))
print(con(n))
```

1.6. Infix , Prefix dan posfix

a. **Infix** adalah notasi yang membentuk atas operator dengan operand,dimana operator berada diantara operand.
   Contoh :
   - A+B
   - A+BxC
   - (AxB)+C

b. **Prefix** adalah notasi yang terbentuk atas operator dengan operand, dimana oprator didepan operand.
   Contoh :
   - A+B
   - A+BxC

   Maka :
   - +AB
   - +AxBC

c. **Postfix** adalah notasi yang membentuk atas operator dengan operand, dimana operator berada dibelakang operand.
   Contoh :
   - A+B
   - A+BxC

Maka :

- AB+
- ABCx+

Contoh Code konversi infix ke postfix:

```
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)')
```