STRUKTUR DATA

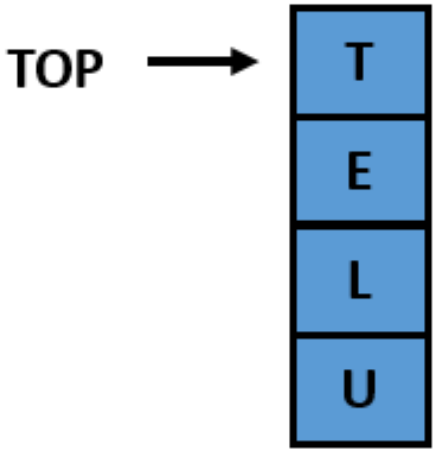
Konsep Dasar Stack

POKOK BAHASAN

- Konsep Dasar Stack
- ADT dan Primitif Stack Representasi List
- ADT dan Primitif Stack Representasi Array
- Konsep Dasar Queue
- ADT dan Primitif Queue Representasi List
- ADT dan Primitif Queue Representasi Array
- Studi Kasus Stack Queue

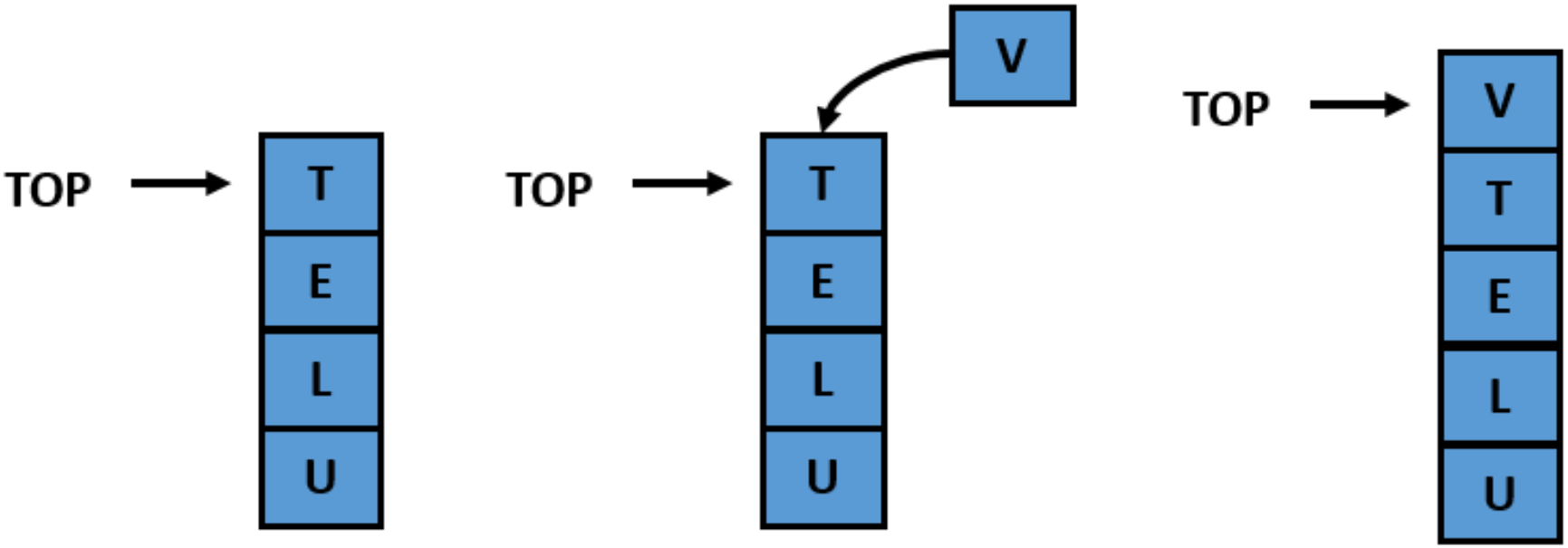
STACK

- Berisi sekumpulan elemen dimana penyisipan dan penghapus elemennya sesuai prinsip Last In First Out



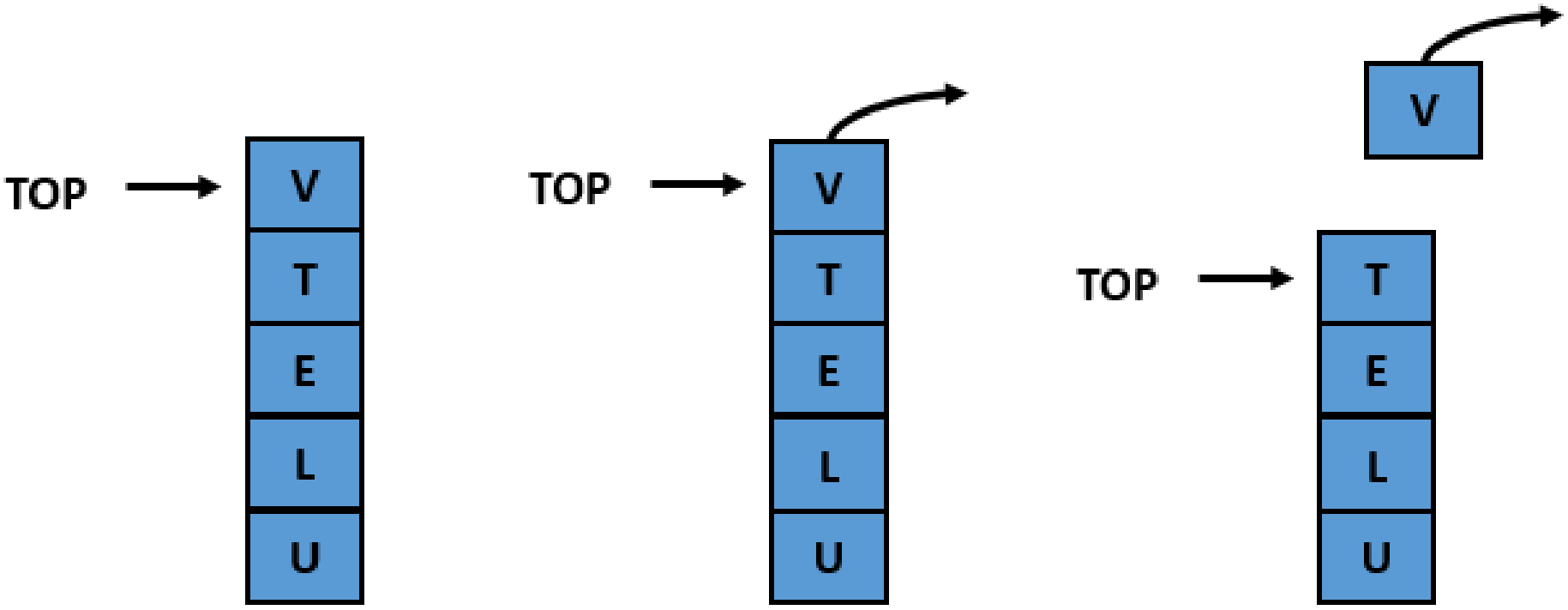
STACK

- Berisi sekumpulan elemen dimana penyisipan dan penghapus elemennya sesuai prinsip Last In First Out



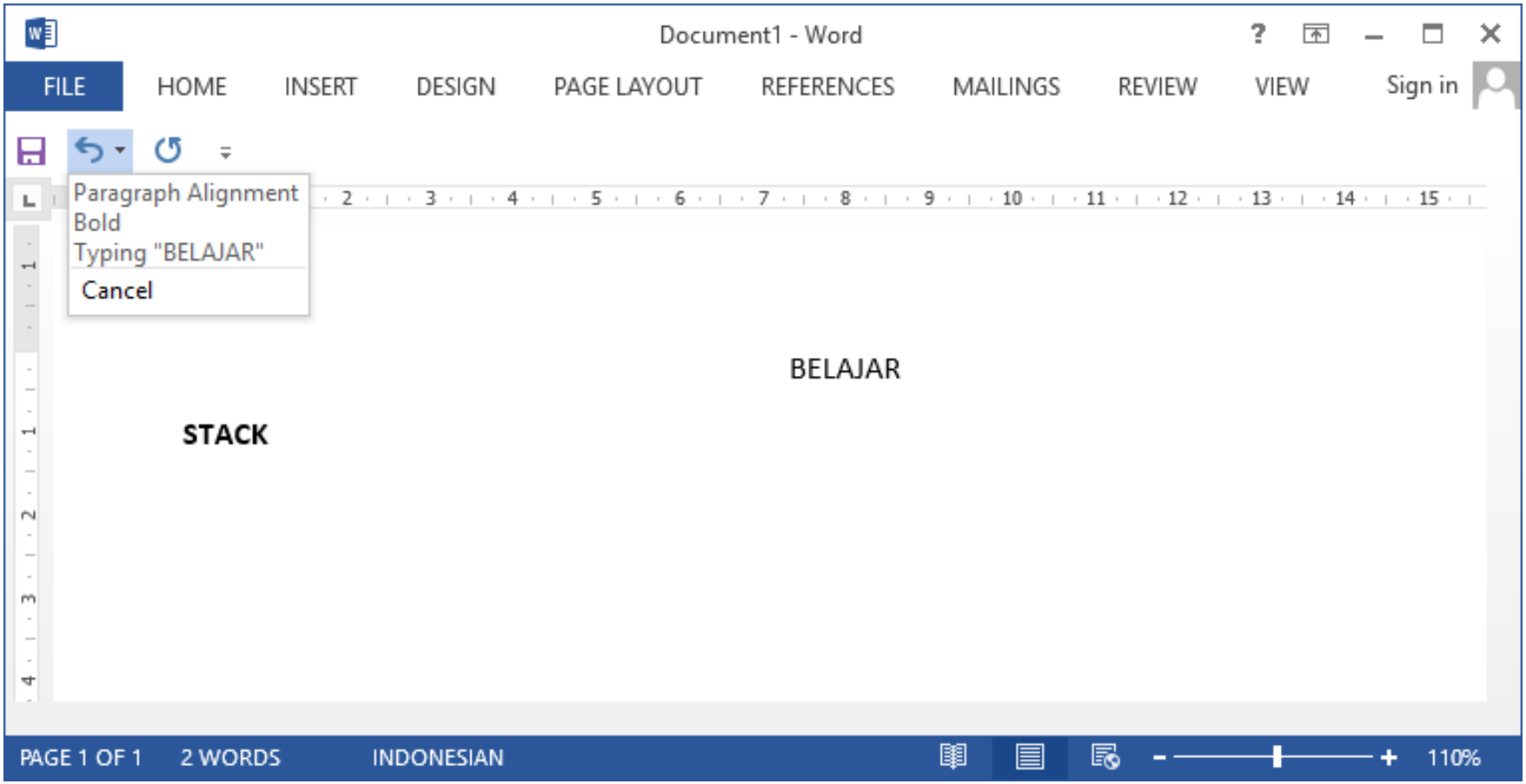
STACK

- Berisi sekumpulan elemen dimana penyisipan dan penghapus elemennya sesuai prinsip Last In First Out



CONTOH STACK DI DUNIA NYATA

CONTOH STACK DI BIDANG KOMPUTER



TERMINOLOGI PADA STACK

STACK
- struktur data dimana penambahan dan penghapusan elemen dilakukan di posisi Top

TOP
- ujung berupa tempat dilakukan penambahan dan penghapusan

PUSH
- perintah untuk menambahkan elemen ke stack

POP
- perintah untuk mengeluarkan elemen dari stack

TERMINOLOGI PADA STACK

ISEMPTY

- perintah untuk memeriksa apakah stack kosong

ISFULL

- perintah untuk memeriksa apakah stack penuh

STRUKTUR DATA

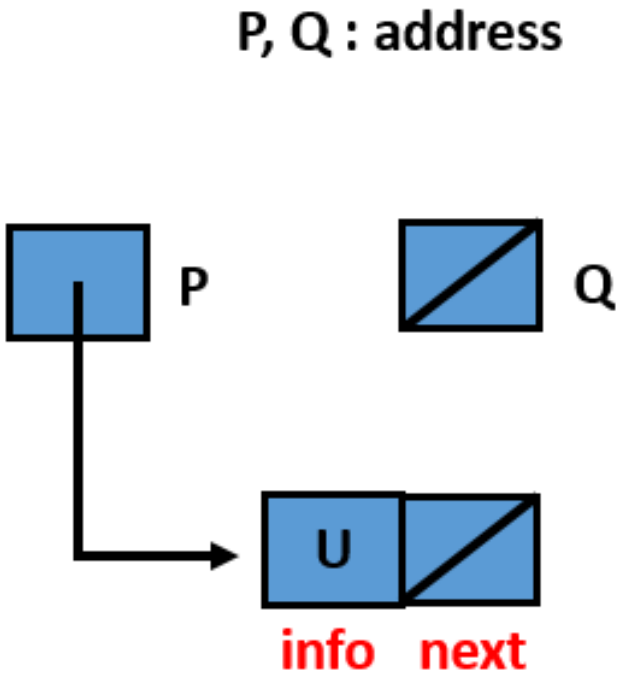
ADT dan Primitif Stack Representasi List

ADT STACK REPRESENTASI LIST

type infotype : **character**
type address : **pointer** **to** ElmtStack

type ElmtStack : <
    info : infotype ,
    next : address
>

type Stack : < Top : address >

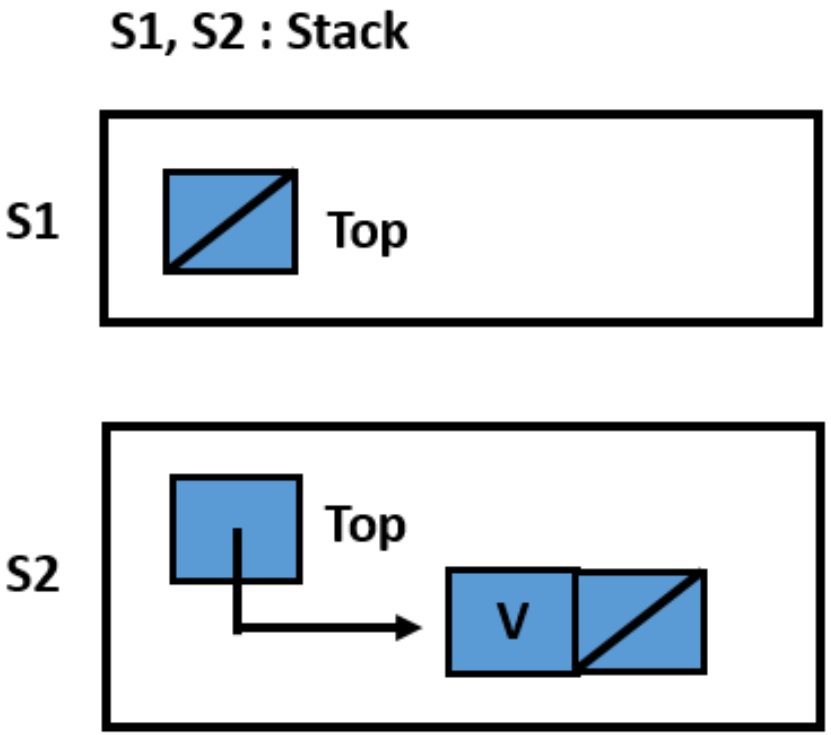


ADT STACK REPRESENTASI LIST

type infotype : **character**
type address : **pointer** **to** ElmtStack

type ElmtStack : <
    info : infotype ,
    next : address
>

type Stack : < Top : address >

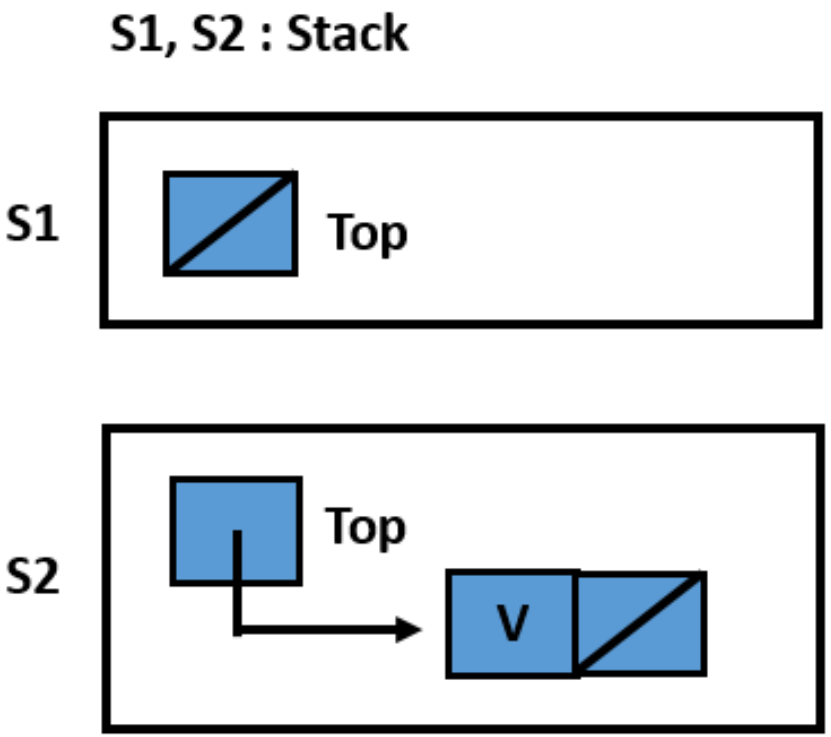


PRIMITIF STACK

- **function** isEmpty (S: Stack) → **boolean**
- **function** createStack () → Stack
- **procedure** Push (**Input/Output** S: Stack, **Input** P: address)
- **procedure** Pop (**Input/Output** S: Stack, **Output** P: address)

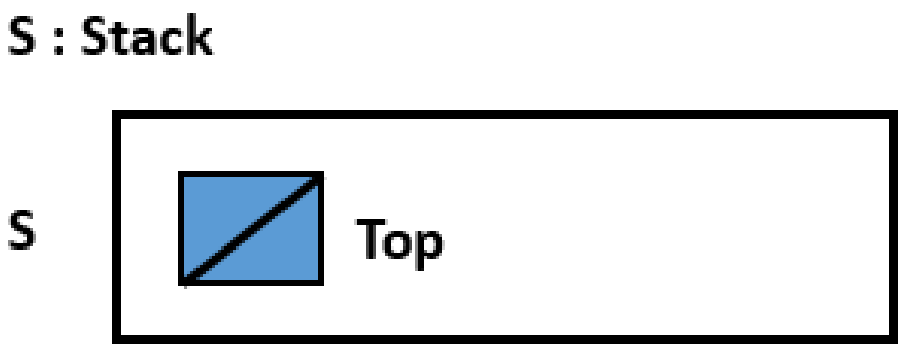
FUNGSI ISEMPTY

1. **function** isEmpty (S : Stack) → **boolean**
2. *{Mengembalikan true jika stack kosong dan false jika tidak}*
3. **kamus lokal**
4. **algoritma**
5. **if** Top(S) = NULL **then**
6. → true
7. **else**
8. → false
9. *{end function}*



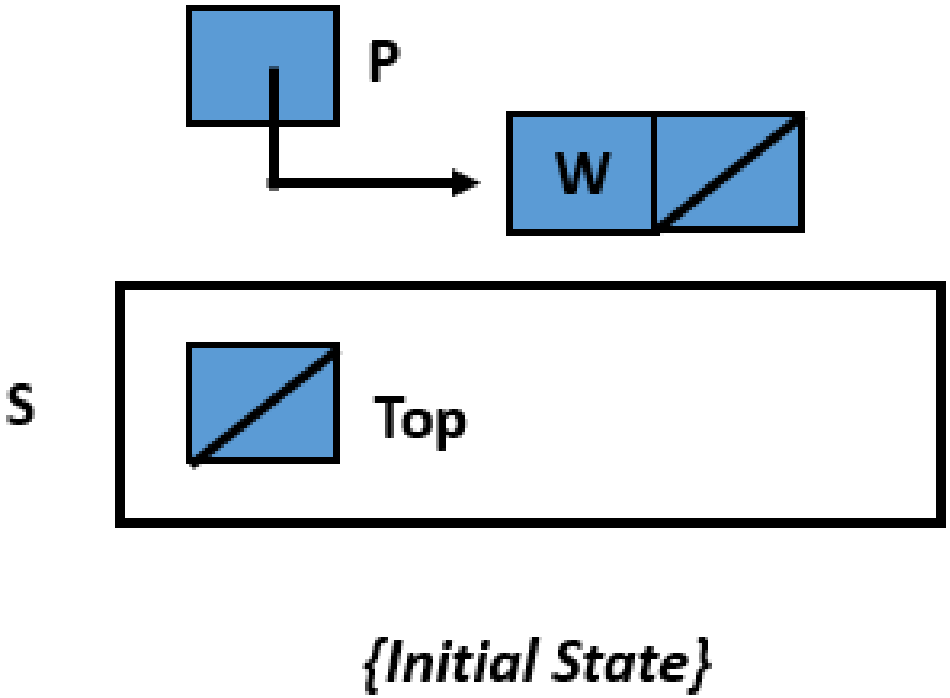
FUNGSI CREATESTACK

1. **function** createStack → Stack
2. *{Mengembalikan stack kosong}*
3. **kamus lokal**
4. S : Stack
5. **algoritma**
6. Top(S) ← NULL
7. → S
8. *{end function}*



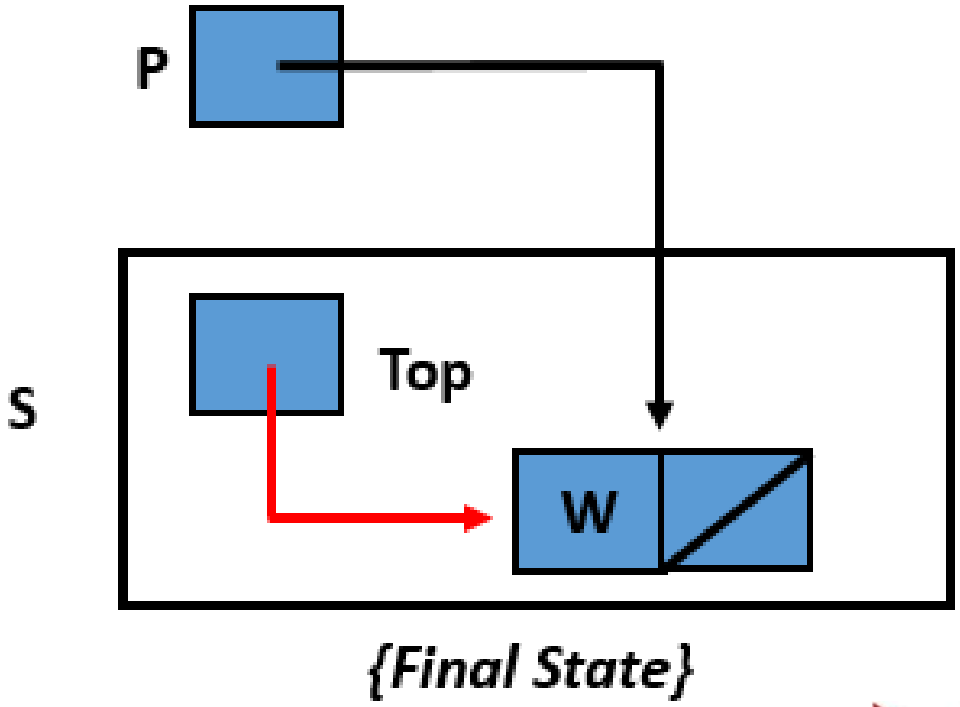
PROSEDUR PUSH

```
1.  procedure  Push (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S mungkin kosong dan elemen di P
3.  F.S. P menjadi elemen di Top S}
4.  kamus lokal

5.  algoritma
6.      if Top(S) = NULL then
7.          Top(S) ← P
8.      else
9.          next(P) ← Top(S)
10.         Top(S)  ← P
11.     {end if}
12. {end procedure}
```



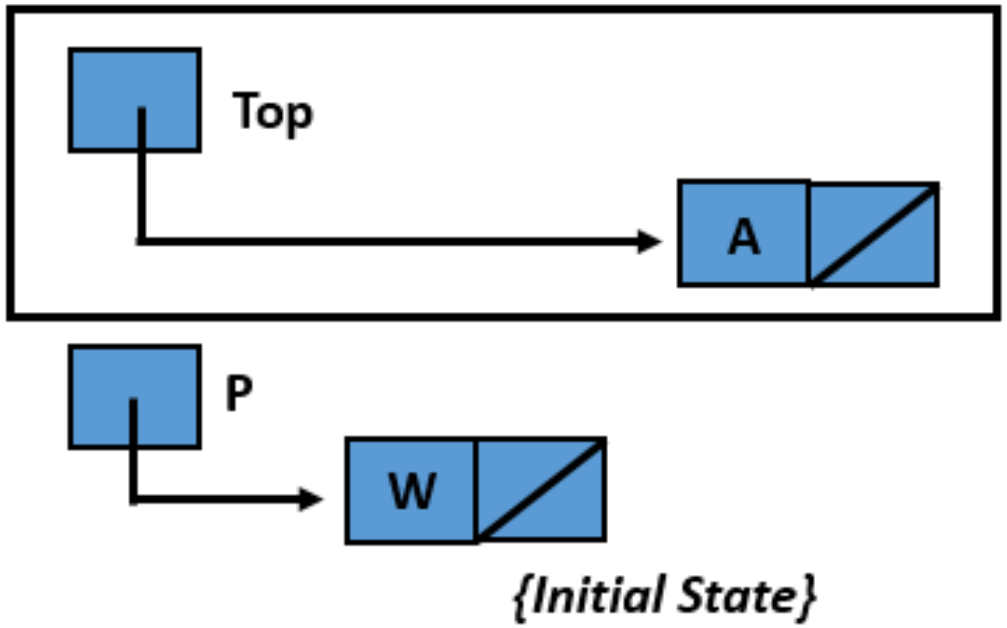
PROSEDUR PUSH

```
1.  procedure  Push (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S mungkin kosong dan elemen di P
3.  F.S. P menjadi elemen di Top S}
4.  kamus lokal

5.  algoritma
6.      if Top(S) = NULL then
7.          Top(S) ← P
8.      else
9.          next(P) ← Top(S)
10.         Top(S)  ← P
11.     {end if}
12. {end procedure}
```



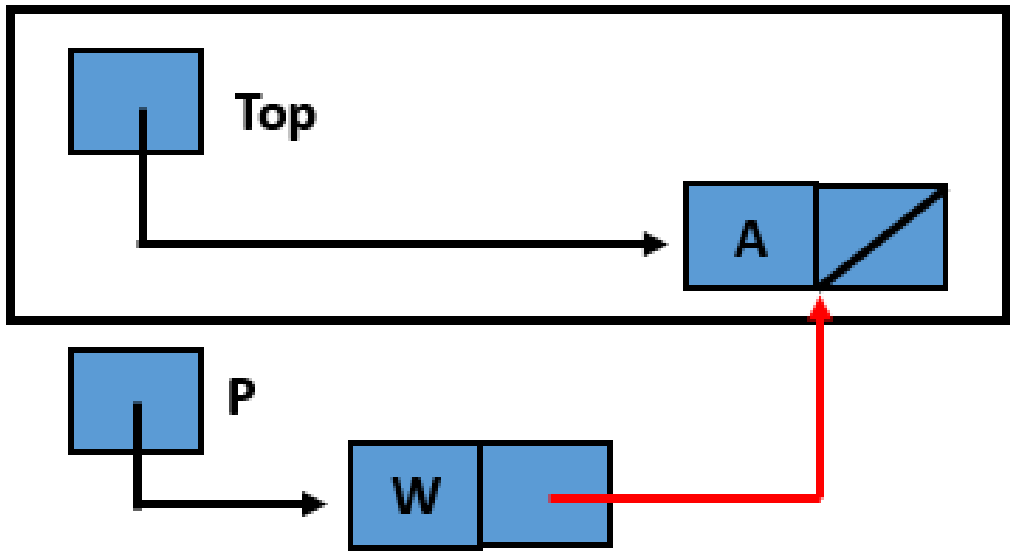
PROSEDUR PUSH

```
1.  procedure Push (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S mungkin kosong dan elemen di P
3.  F.S. P menjadi elemen di Top S}
4.  kamus lokal

5.  algoritma                          S
6.      if Top(S) = NULL then
7.          Top(S) ← P
8.      else
9.          next(P) ← Top(S)
10.         Top(S)  ← P
11.     {end if}
12. {end procedure}
```



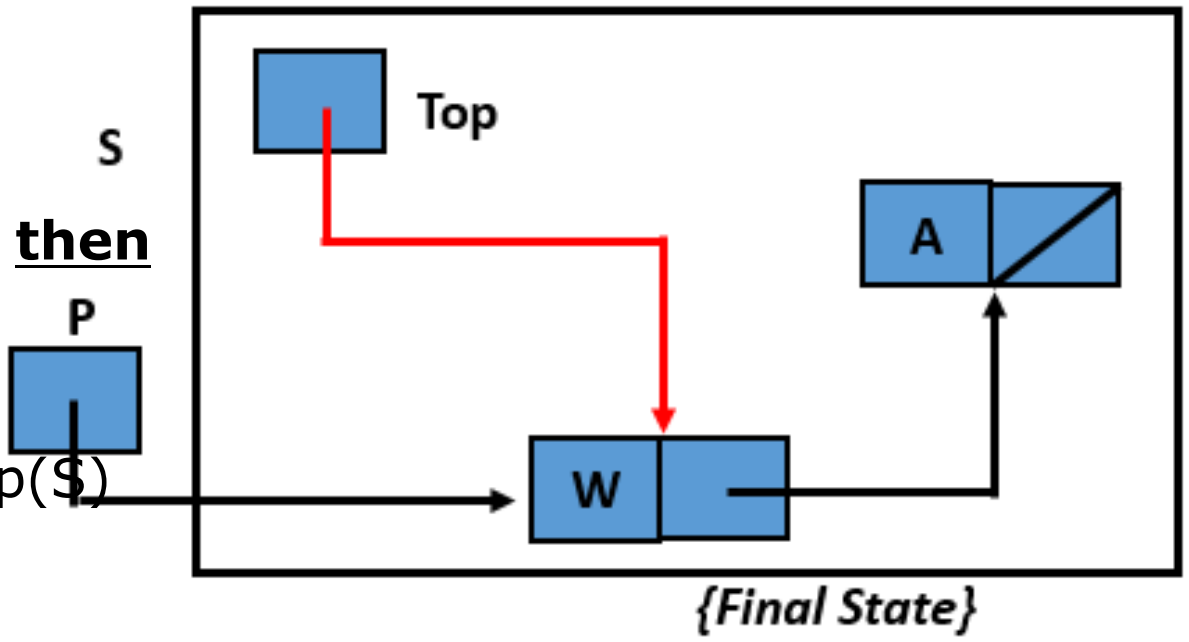
PROSEDUR PUSH

1. **procedure** Push (**Input/Output** S: Stack, **Input** P: address)
2. *{I.S. Terdefinisi stack S mungkin kosong dan elemen di P*
3. *F.S. P menjadi elemen di Top S}*
4. **kamus lokal**
5. **algoritma** S
6. **if** Top(S) = NULL **then**
7. Top(S) ← P
8. **else**
9. next(P) ← Top(S)
10. Top(S) ← P
11. *{end if}*
12. *{end procedure}*



PROSEDUR PUSH

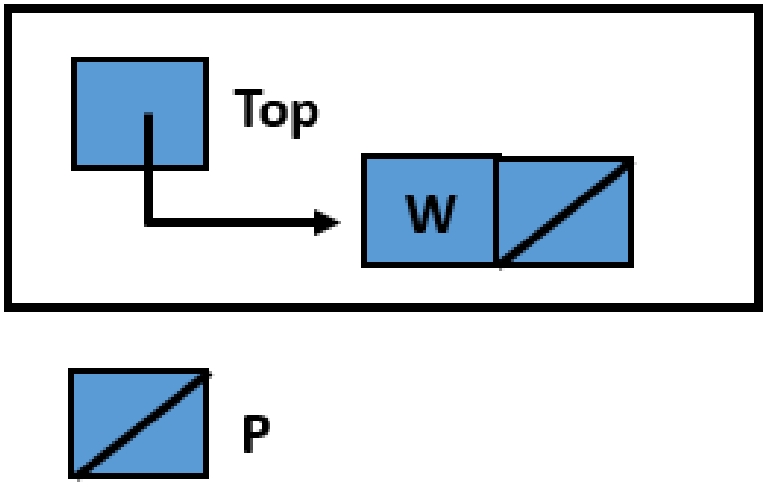
```
1.  procedure Push (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S mungkin kosong dan elemen di P
3.  F.S. P menjadi elemen di Top S}
4.  kamus lokal

5.  algoritma
6.      if Top(S) = NULL then
7.          Top(S) ← P
8.      else
9.          next(P) ← Top(S)
10.         Top(S) ← P
11.     {end if}
12. {end procedure}
```



PROSEDUR POP

```
1.  procedure  Pop (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S tidak kosong
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong}
4.  kamus lokal
5.  algoritma
6.      if next(Top(S)) = NULL then
7.          P ← Top(S)
8.          Top(S) ← NULL
9.      else
10.         P ← Top(S)
11.         Top(S) ← next(Top(S))
12.         next(P) ← NULL
13.     {end if}
14. {end procedure}
```

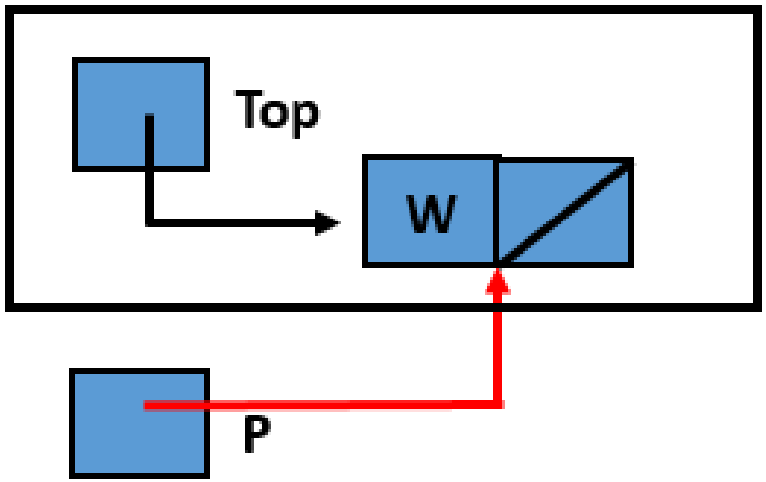
S



{Initial State}

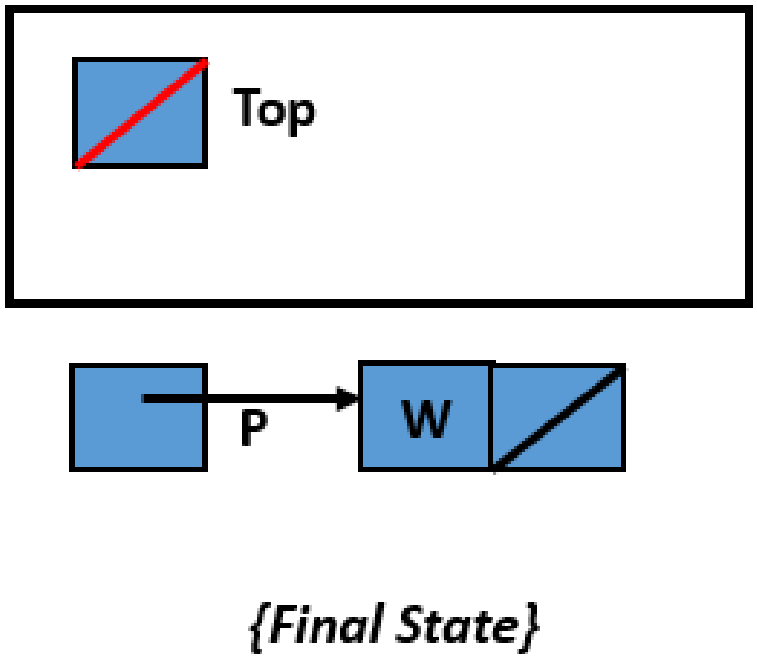
PROSEDUR POP

```
1.  procedure  Pop (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S tidak kosong
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong}
4.  kamus lokal
5.  algoritma
6.      if next(Top(S)) = NULL then
7.          P ← Top(S)
8.          Top(S) ← NULL
9.      else
10.         P ← Top(S)
11.         Top(S) ← next(Top(S))
12.         next(P) ← NULL
13.     {end if}
14. {end procedure}
```



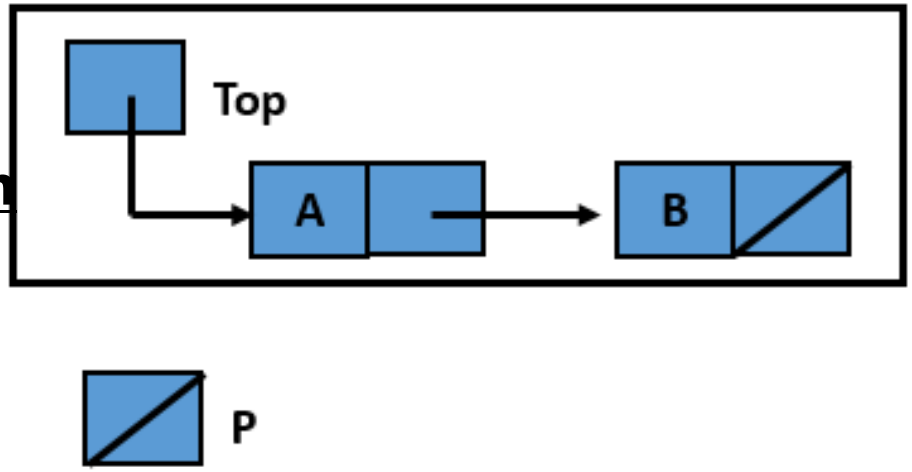
PROSEDUR POP

```
1.  procedure  Pop (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S tidak kosong
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong}
4.  kamus lokal
5.  algoritma
6.      if next(Top(S)) = NULL then
7.          P ← Top(S)
8.          Top(S) ← NULL
9.      else
10.         P ← Top(S)
11.         Top(S) ← next(Top(S))
12.         next(P) ← NULL
13.     {end if}
14. {end procedure}
```



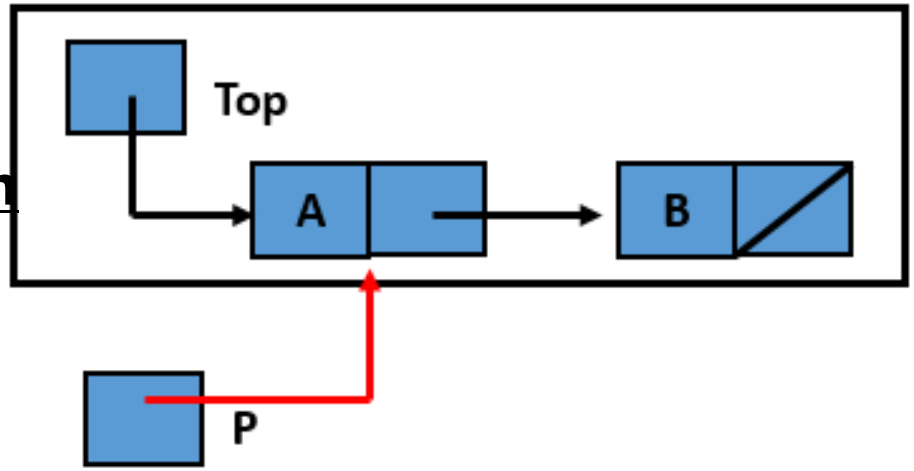
PROSEDUR POP

```
1.  procedure  Pop (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S tidak kosong
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong}
4.  kamus lokal
5.  algoritma
6.     if next(Top(S)) = NULL then
7.           P ← Top(S)
8.           Top(S) ← NULL
9.     else
10.          P ← Top(S)
11.          Top(S) ← next(Top(S))
12.          next(P) ← NULL
13.    {end if}
14. {end procedure}
```



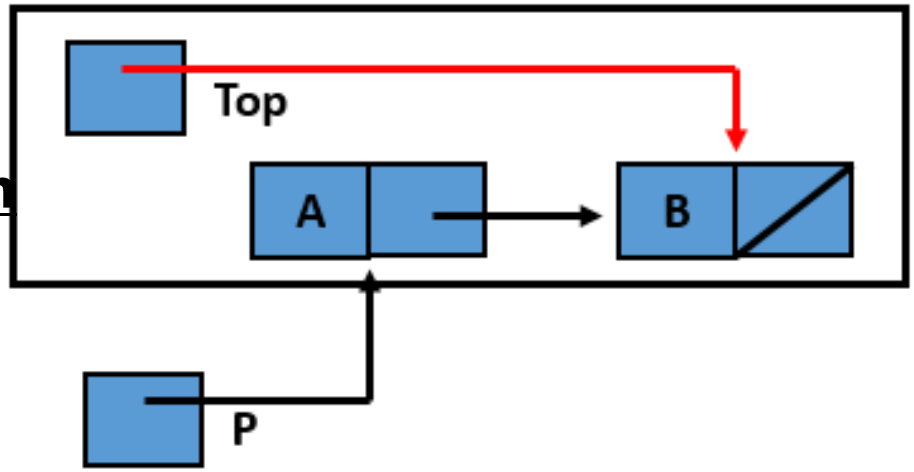
PROSEDUR POP

```
1.  procedure  Pop (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S tidak kosong
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong}
4.  kamus lokal
5.  algoritma
6.     if next(Top(S)) = NULL then
7.           P ← Top(S)
8.           Top(S) ← NULL
9.     else
10.          P ← Top(S)
11.          Top(S) ← next(Top(S))
12.          next(P) ← NULL
13.    {end if}
14. {end procedure}
```



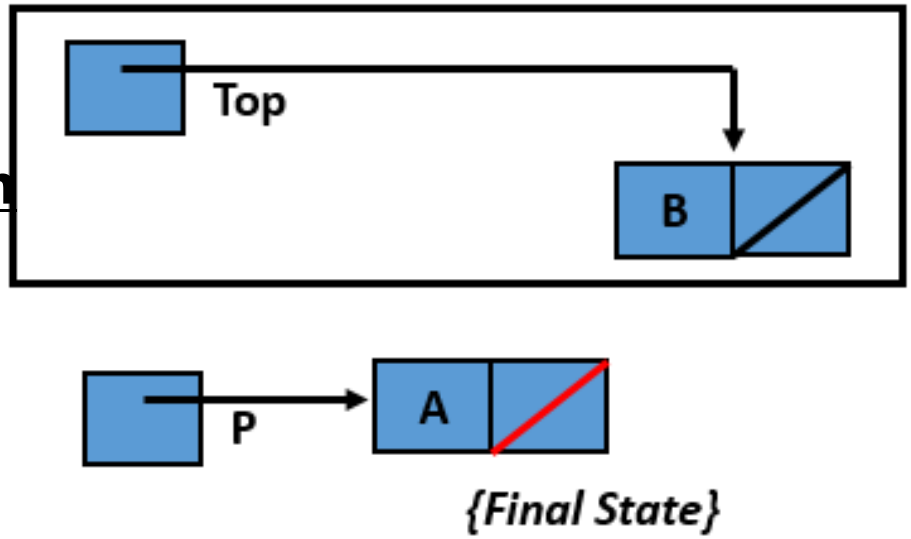
PROSEDUR POP

```
1.  procedure  Pop (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S tidak kosong
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong}
4.  kamus lokal
5.  algoritma
6.     if next(Top(S)) = NULL then
7.           P ← Top(S)
8.           Top(S) ← NULL
9.     else
10.          P ← Top(S)
11.          Top(S) ← next(Top(S))
12.          next(P) ← NULL
13.    {end if}
14. {end procedure}
```



PROSEDUR POP

```
1.  procedure  Pop (Input/Output S: Stack, Input P: address)
2.  {I.S. Terdefinisi stack S tidak kosong
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong}
4.  kamus lokal
5.  algoritma
6.     if next(Top(S)) = NULL then
7.           P ← Top(S)
8.           Top(S) ← NULL
9.     else
10.          P ← Top(S)
11.          Top(S) ← next(Top(S))
12.          next(P) ← NULL
13.    {end if}
14. {end procedure}
```

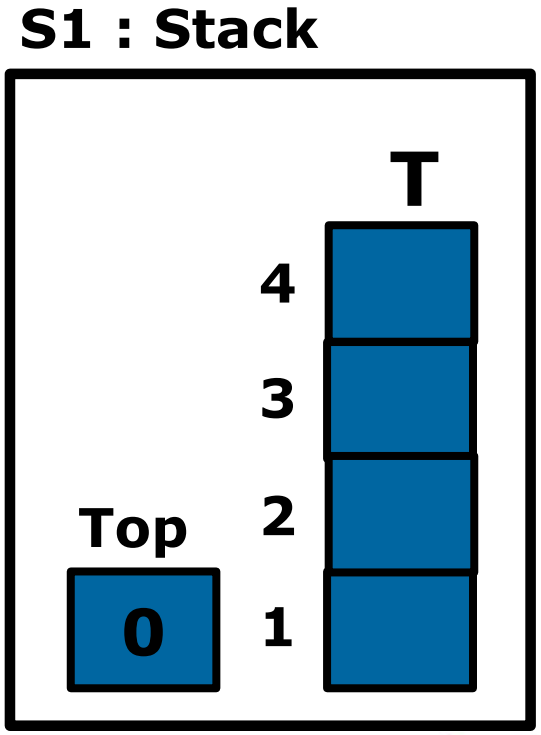


STRUKTUR DATA

ADT dan Primitif Stack Representasi Array

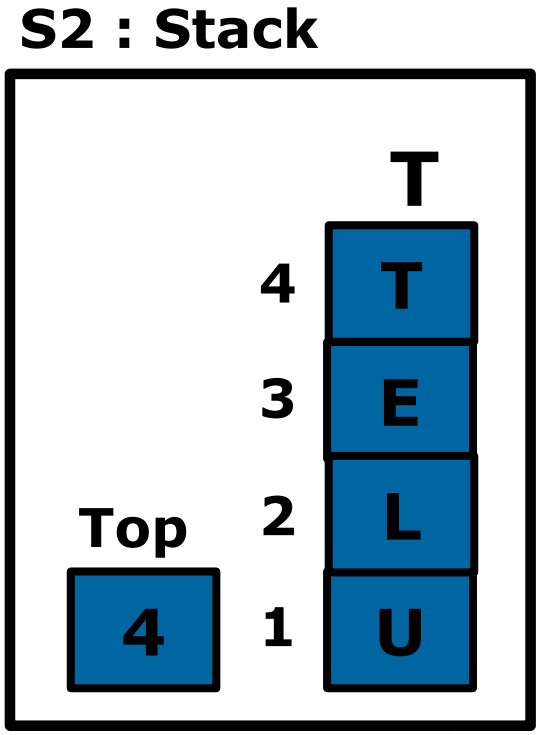
ADT STACK REPRESENTASI ARRAY

```
constant  IDXMAX : integer = 4
type  infotype :  character
type  address :  integer
type  Stack : <
    T      : array  [1..IDXMAX]  of  infotype,
    Top    : address
>
```



ADT STACK REPRESENTASI ARRAY

```
constant  IDXMAX : integer = 4
type  infotype :  character
type  address :  integer
type  Stack : <
    T      : array  [1..IDXMAX]  of  infotype,
    Top    : address
>
```

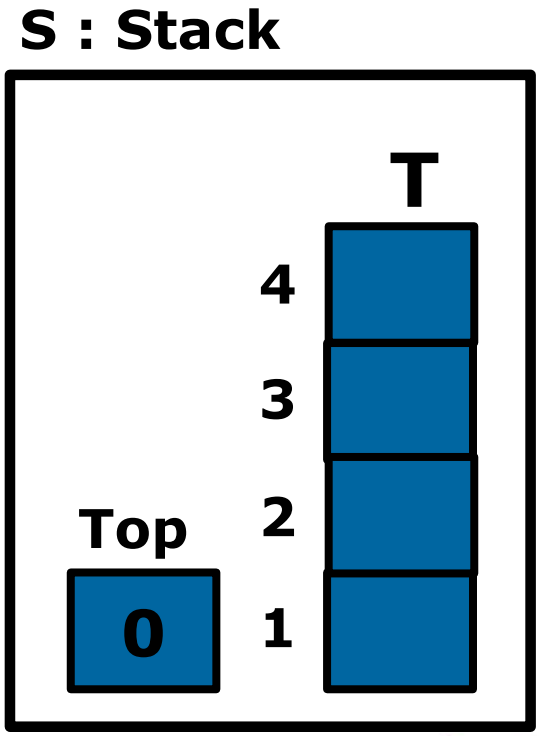


PRIMITIF STACK

- **function** isEmpty (S : Stack) → **boolean**
- **function** isFull (S : Stack) → **boolean**
- **function** createStack () → Stack
- **procedure** Push (**Input/Output** S: Stack, **Input** P: infotype)
- **procedure** Pop (**Input/Output** S: Stack, **Output** P: infotype)

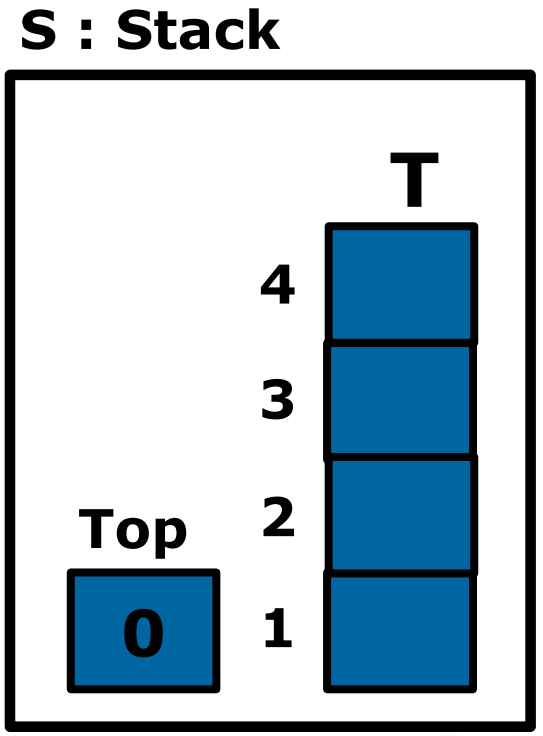
ILUSTRASI PRIMITIF STACK

1. S ← createStack()



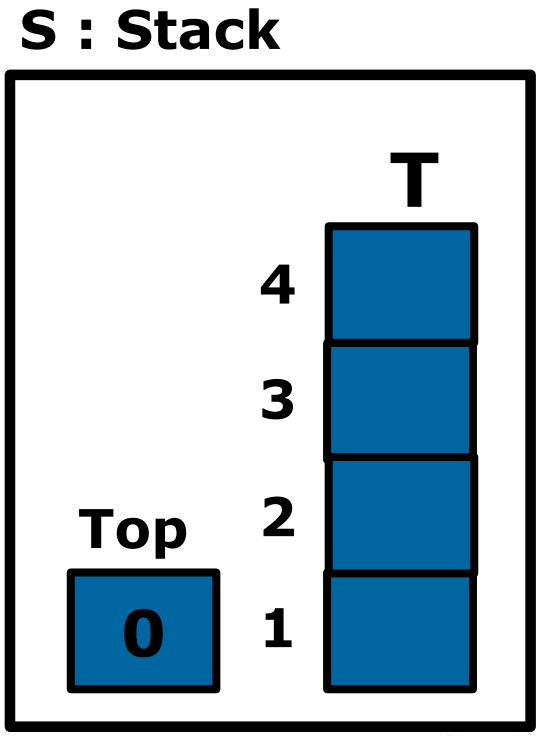
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*



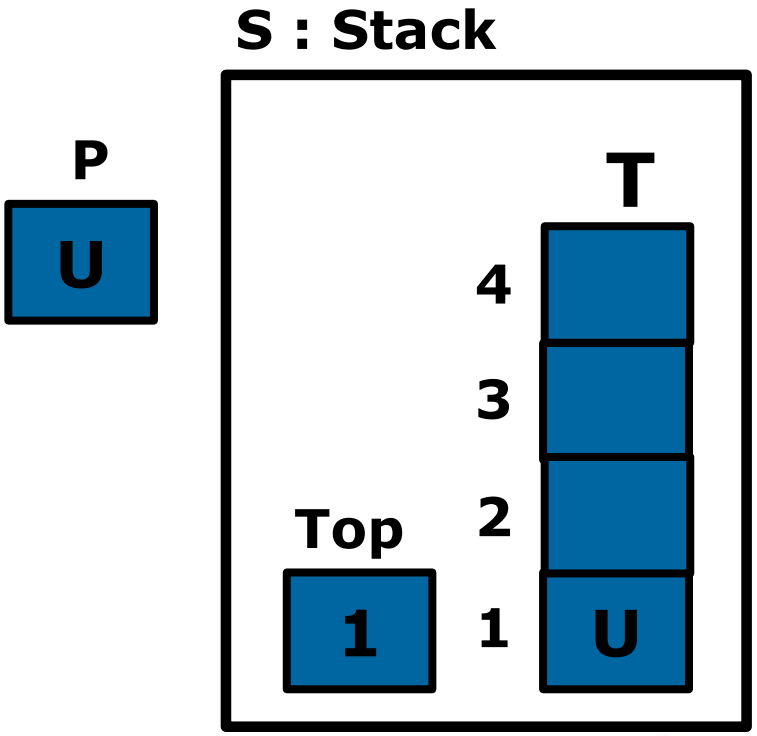
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*



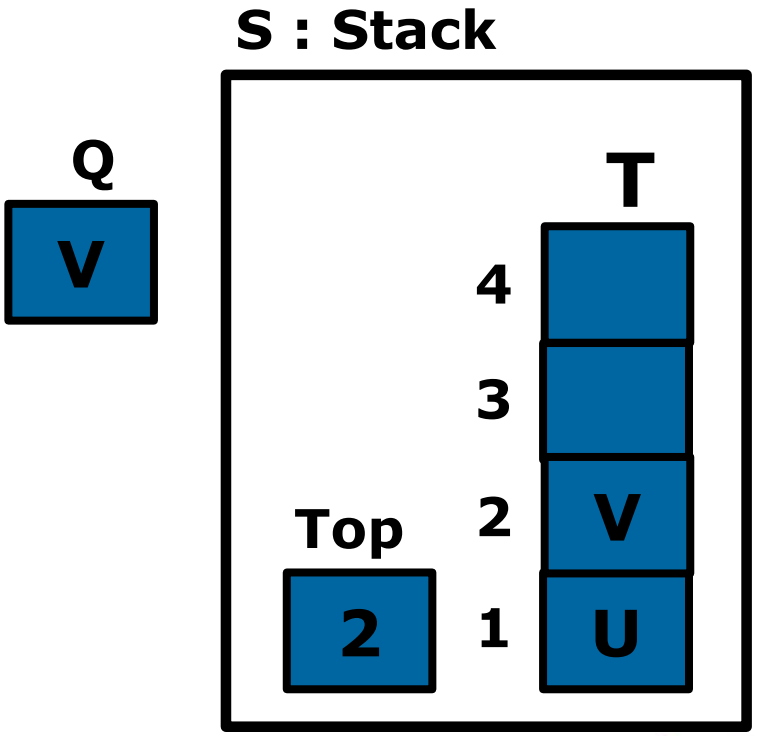
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*
4. Push(S, P)



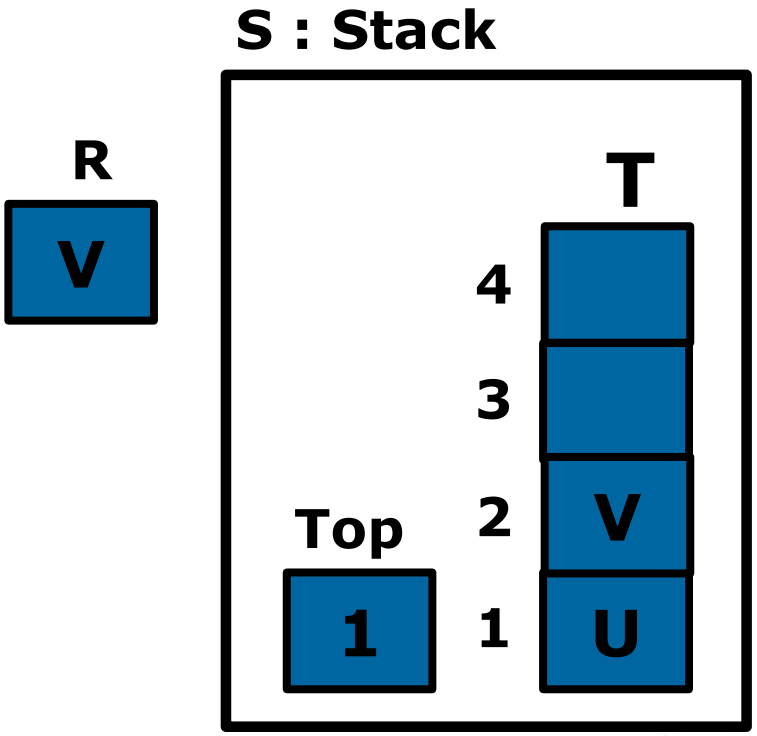
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*
4. Push(S, P)
5. Push(S, Q)



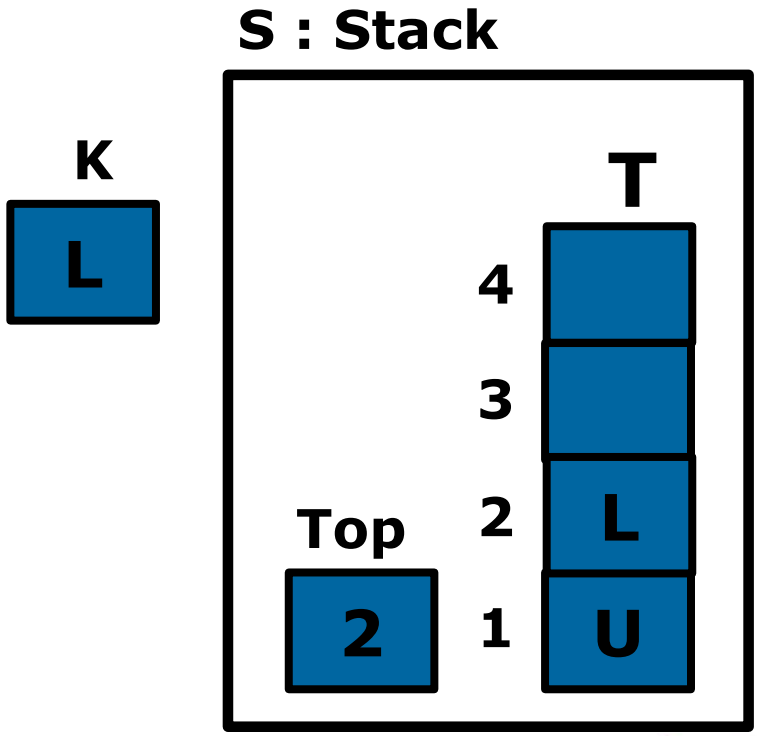
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*
4. Push(S, P)
5. Push(S, Q)
6. Pop(S, R)



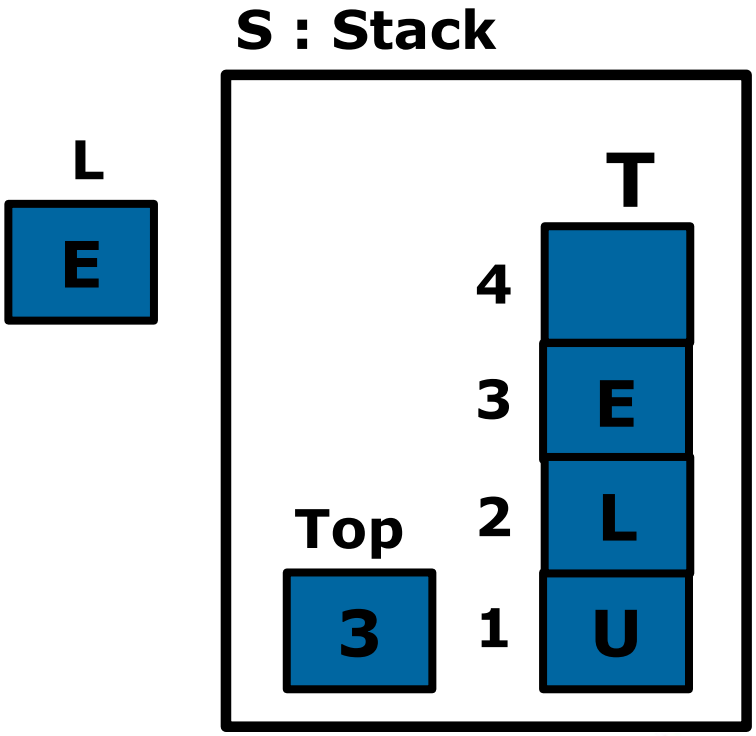
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*
4. Push(S, P)
5. Push(S, Q)
6. Pop(S, R)
7. Push(S, K)



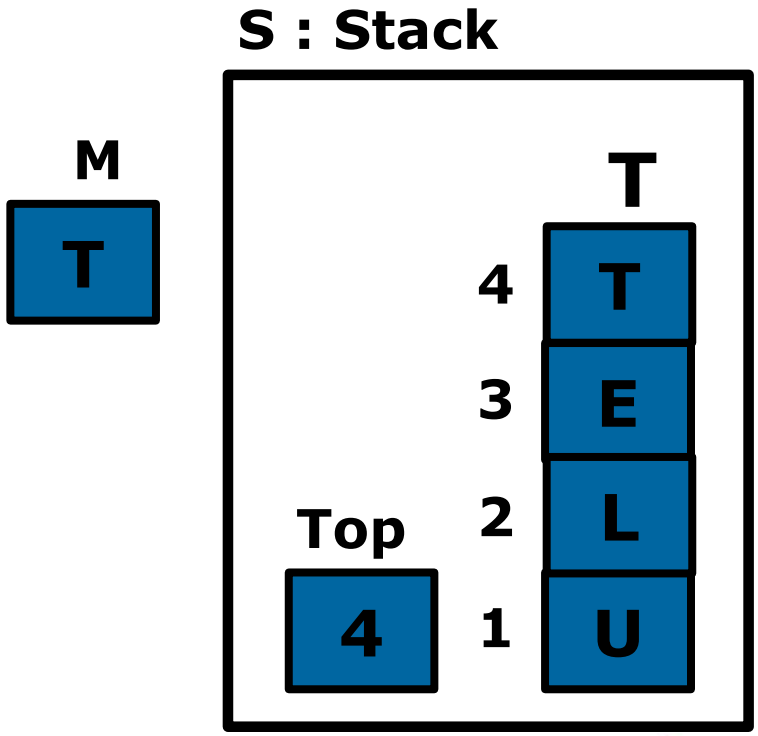
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*
4. Push(S, P)
5. Push(S, Q)
6. Pop(S, R)
7. Push(S, K)
8. Push(S, L)



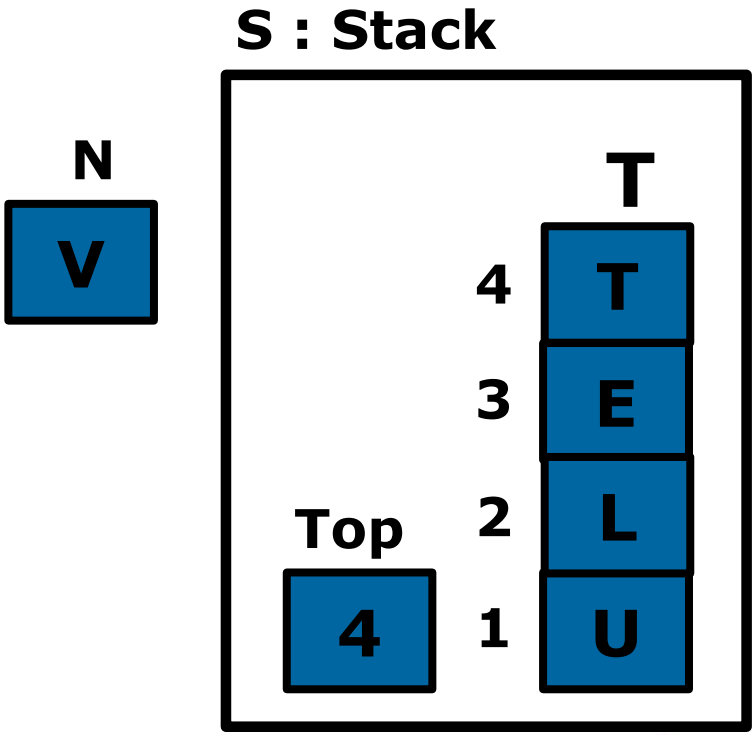
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*
4. Push(S, P)
5. Push(S, Q)
6. Pop(S, R)
7. Push(S, K)
8. Push(S, L)
9. Push(S, M)



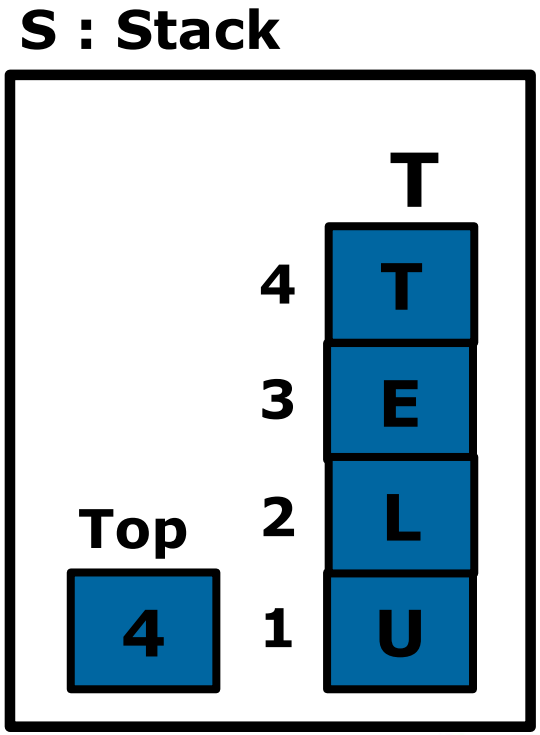
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*
4. Push(S, P)
5. Push(S, Q)
6. Pop(S, R)
7. Push(S, K)
8. Push(S, L)
9. Push(S, M)
10. Push(S, N) *{gagal}*



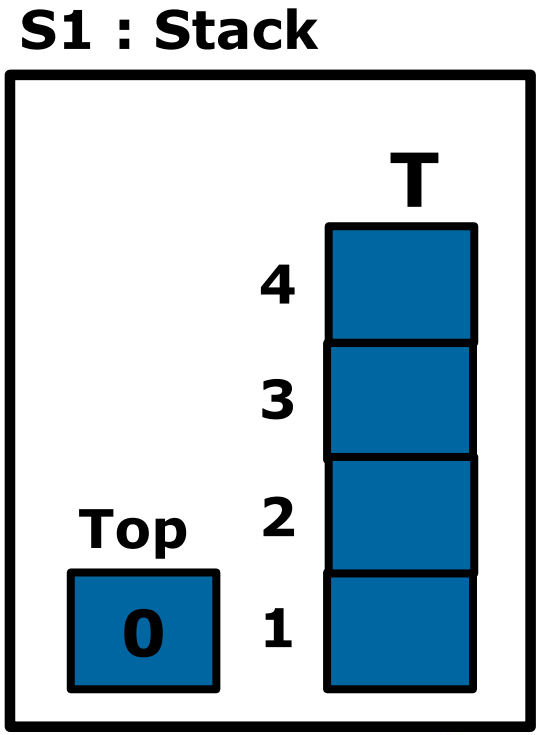
ILUSTRASI PRIMITIF STACK

1. S ← createStack()
2. isEmpty(S) *{return true}*
3. isFull(S) *{return false}*
4. Push(S, P)
5. Push(S, Q)
6. Pop(S, R)
7. Push(S, K)
8. Push(S, L)
9. Push(S, M)
10. Push(S, N) *{gagal}*
11. isFull(S) *{return true}*



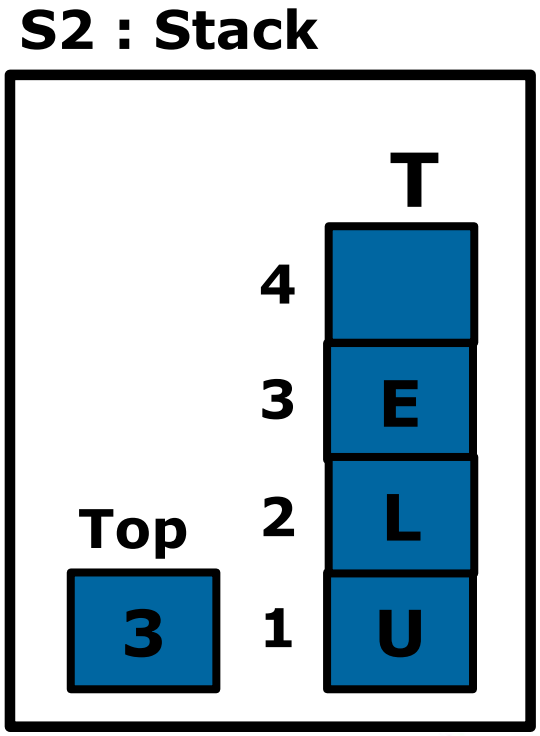
FUNGSI ISEMPTY

1. **function** isEmpty (S : Stack) → **boolean**
2. *{Mengembalikan true jika stack kosong dan false jika tidak}*
3. **kamus lokal**
4. **algoritma**
5. **if** S.Top = 0 **then**
6. → true
7. **else**
8. → false
9. *{end function}*



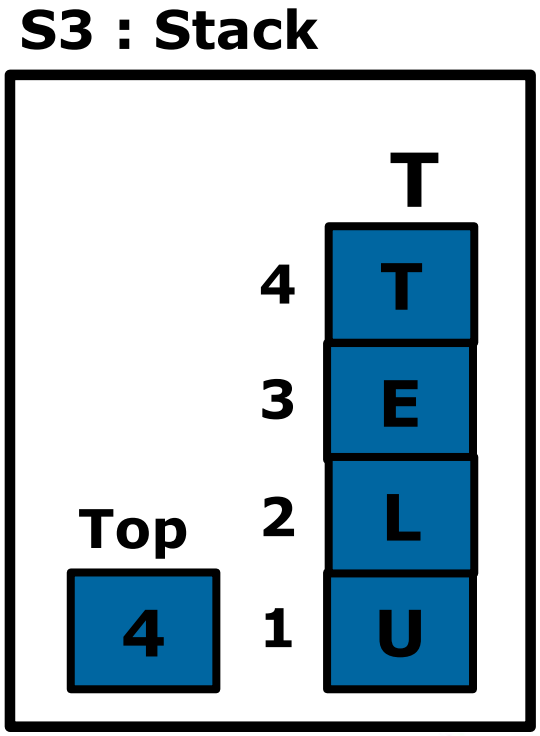
FUNGSI ISEMPTY

1. **function** isEmpty (S : Stack) → **boolean**
2. *{Mengembalikan true jika stack kosong dan false jika tidak}*
3. **kamus lokal**
4. **algoritma**
5. **if** S.Top = 0 **then**
6. → true
7. **else**
8. → false
9. *{end function}*



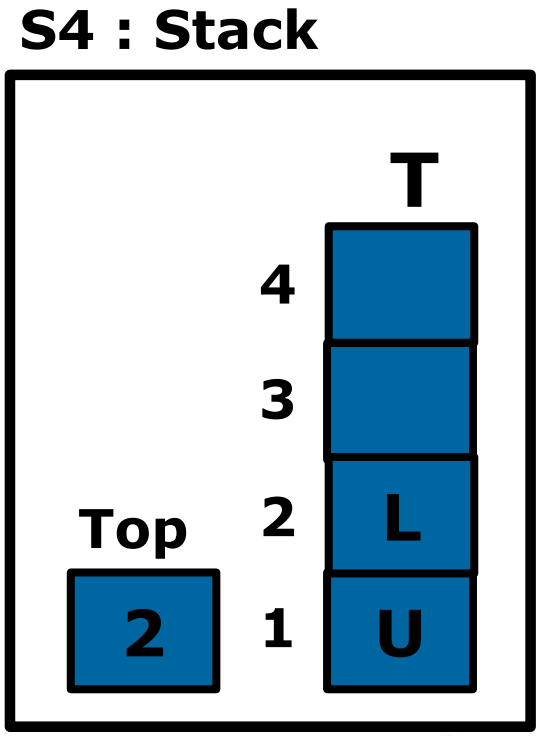
FUNGSI ISFULL

1. **function** isFull (S : Stack) → **boolean**
2. *{Mengembalikan true jika stack penuh dan false jika tidak}*
3. **kamus lokal**
4. **algoritma**
5. **if** S.Top = IDXMAX **then**
6. → true
7. **else**
8. → false
9. *{end function}*



FUNGSI ISFULL

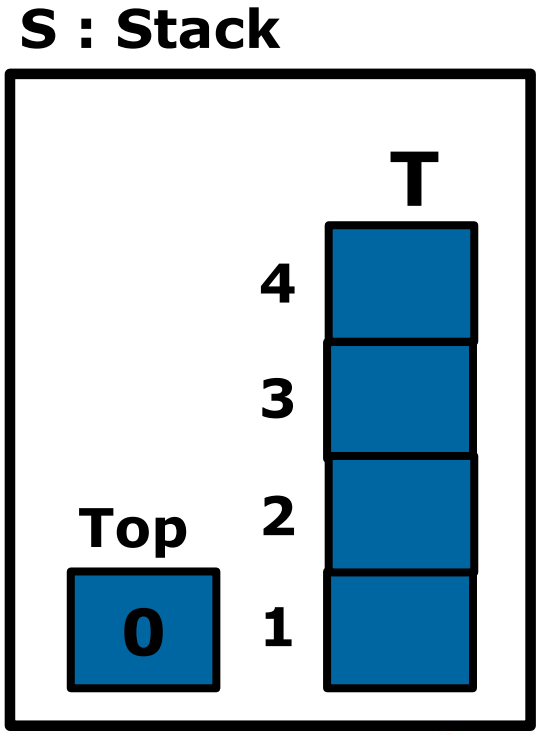
1. **function** isFull (S : Stack) → **boolean**
2. *{Mengembalikan true jika stack penuh dan false jika tidak}*
3. **kamus lokal**
4. **algoritma**
5. **if** S.Top = IDXMAX **then**
6. → true
7. **else**
8. → false
9. *{end function}*



FUNGSI CREATESTACK

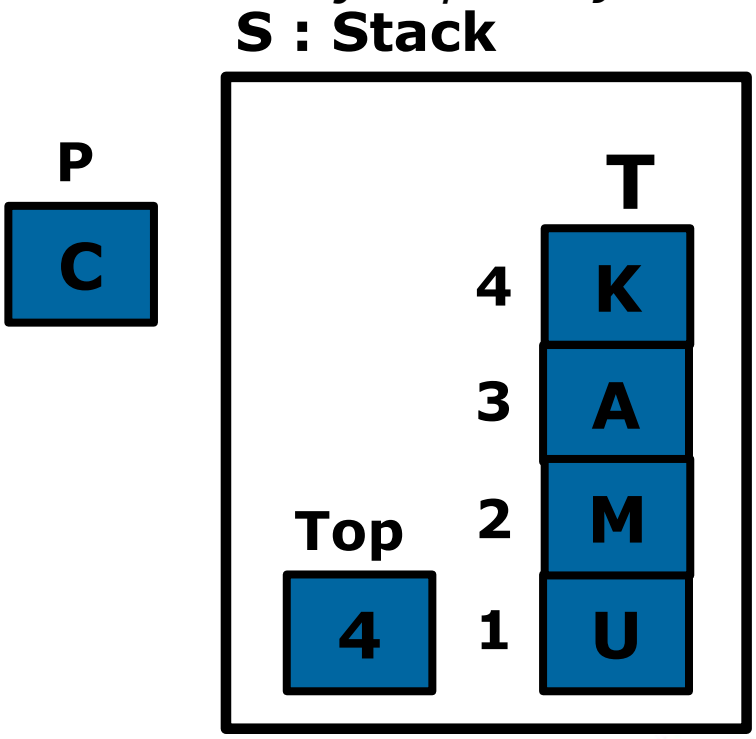
1. **function** createStack → Stack
2. *{Mengembalikan stack kosong}*
3. **kamus lokal**
4. S : Stack
5. **algoritma**
6. S.Top ← 0
7. → S
8. *{end function}*

{Final State}



PROSEDUR PUSH

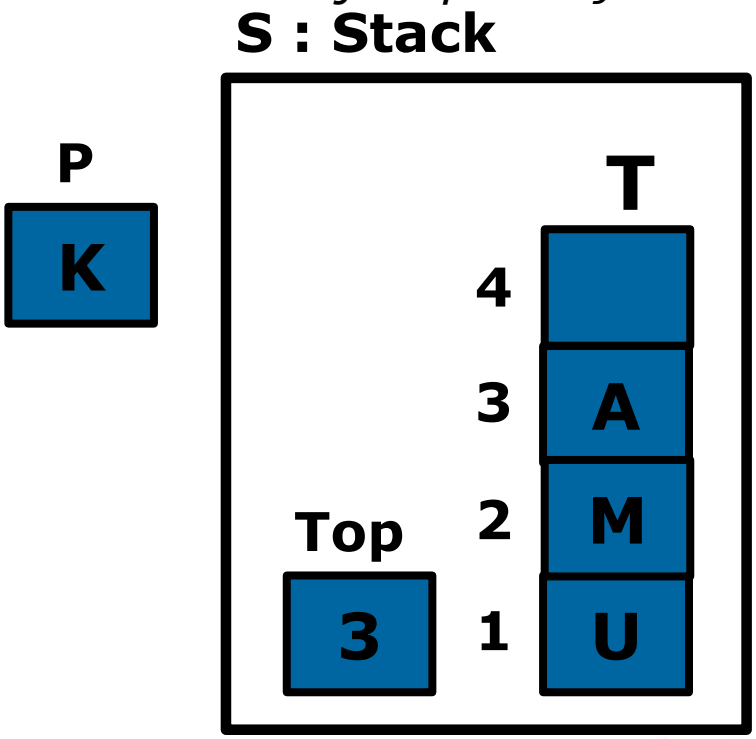
```
1.  procedure Push (Input/Output S : Stack, Input P : infotype)
2.  {I.S. Terdefinisi stack S mungkin kosong/penuh dan infotype di P
3.  F.S. P menjadi elemen di Top S jika berhasil. IS=FS jika penuh}
4.  kamus lokal
5.  algoritma
6.      if isFull(S) then
7.          output('Stack penuh')
8.      else
9.          S.Top ← S.Top + 1
10.         S.T[Top] ← P
11.     {end if}
12. {end procedure}
```



{Initial State = Final State}

PROSEDUR PUSH

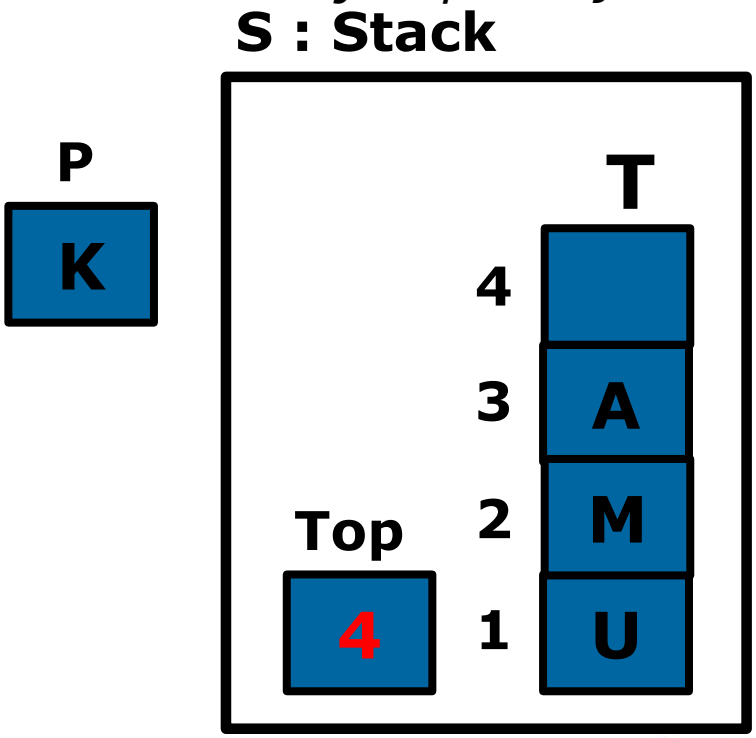
```
1.  procedure  Push (Input/Output S : Stack, Input P : infotype)
2.  {I.S. Terdefinisi stack S mungkin kosong/penuh dan infotype di P
3.  F.S. P menjadi elemen di Top S jika berhasil. IS=FS jika penuh}
4.  kamus lokal
5.  algoritma
6.      if  isFull(S)  then
7.          output('Stack penuh')
8.      else
9.          S.Top ← S.Top + 1
10.         S.T[Top] ← P
11.     {end if}
12. {end procedure}
```



{Initial State}

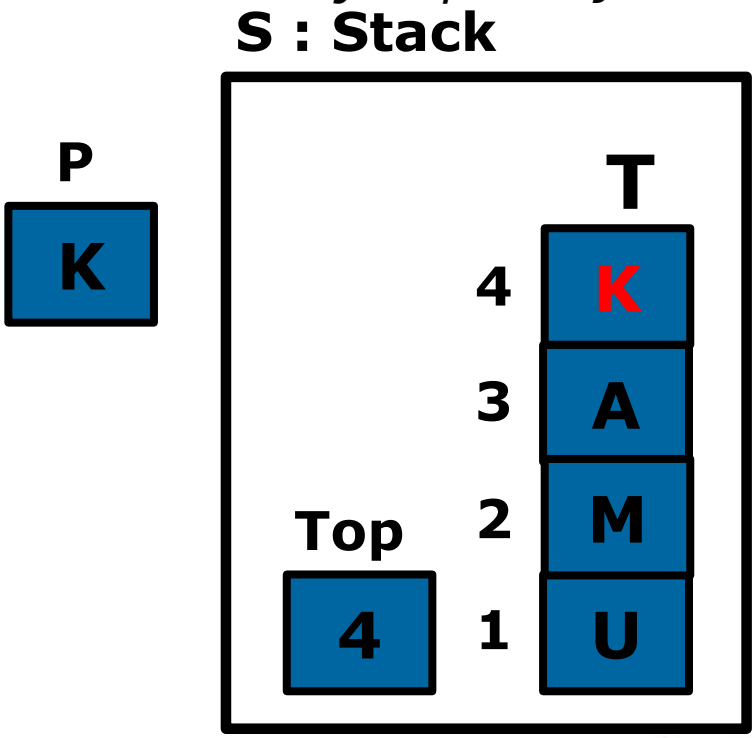
PROSEDUR PUSH

```
1.  procedure  Push (Input/Output S : Stack, Input P : infotype)
2.  {I.S. Terdefinisi stack S mungkin kosong/penuh dan infotype di P
3.  F.S. P menjadi elemen di Top S jika berhasil. IS=FS jika penuh}
4.  kamus lokal
5.  algoritma
6.      if  isFull(S)  then
7.          output('Stack penuh')
8.      else
9.          S.Top ← S.Top + 1
10.         S.T[Top] ← P
11.     {end if}
12. {end procedure}
```



PROSEDUR PUSH

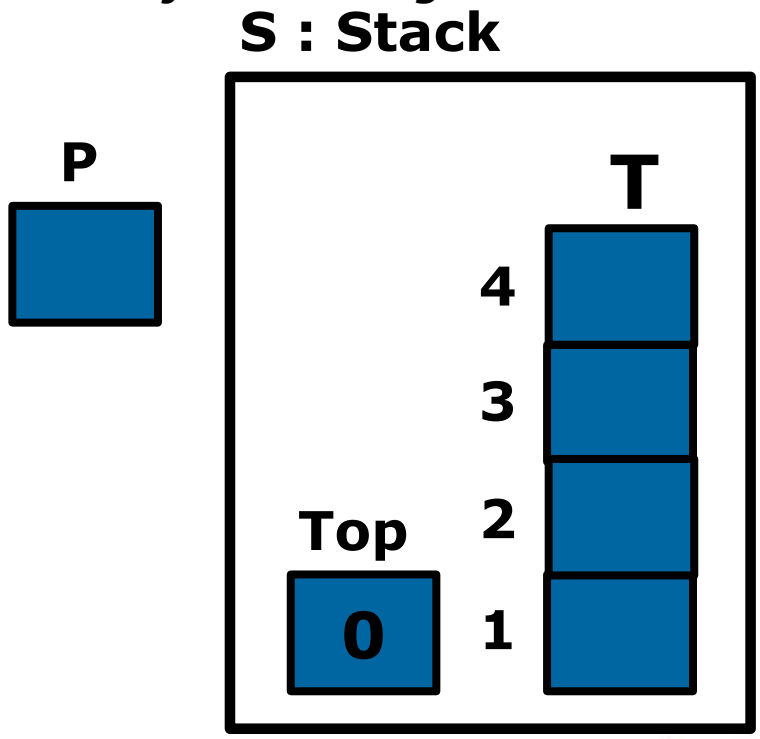
```
1.  procedure  Push (Input/Output S : Stack, Input P : infotype)
2.  {I.S. Terdefinisi stack S mungkin kosong/penuh dan infotype di P
3.  F.S. P menjadi elemen di Top S jika berhasil. IS=FS jika penuh}
4.  kamus lokal
5.  algoritma
6.      if  isFull(S)  then
7.          output('Stack penuh')
8.      else
9.          S.Top ← S.Top + 1
10.         S.T[Top] ← P
11.     {end if}
12. {end procedure}
```



{Final State}

PROSEDUR POP

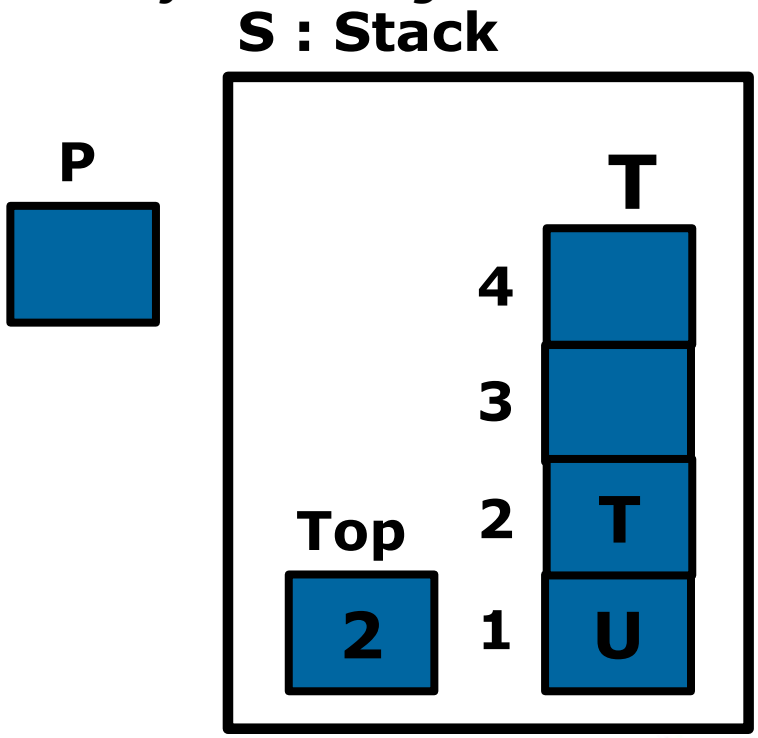
```
1.  procedure  Pop (Input/Output S : Stack, Output P : infotype)
2.  {I.S. Terdefinisi stack S mungkin kosong/penuh
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong.
4.  IS=FS jika stack kosong}
5.  kamus lokal
6.  algoritma
7.     if  isEmpty(S)  then
8.        output('Stack kosong')
9.     else
10.       P ← S.T[Top]
11.       S.Top ← S.Top – 1
12.    {end if}
13. {end procedure}
```



{Initial State = Final State}

PROSEDUR POP

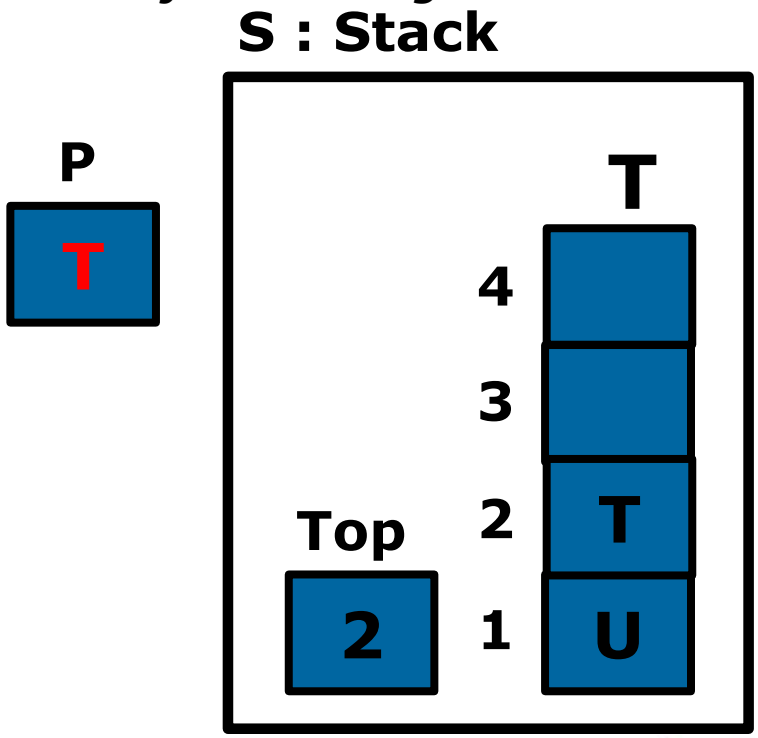
```
1.  procedure  Pop (Input/Output S : Stack, Output P : infotype)
2.  {I.S. Terdefinisi stack S mungkin kosong/penuh
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong.
4.  IS=FS jika stack kosong}
5.  kamus lokal
6.  algoritma
7.     if  isEmpty(S)  then
8.        output('Stack kosong')
9.     else
10.       P ← S.T[Top]
11.       S.Top ← S.Top – 1
12.    {end if}
13. {end procedure}
```



{Initial State}

PROSEDUR POP

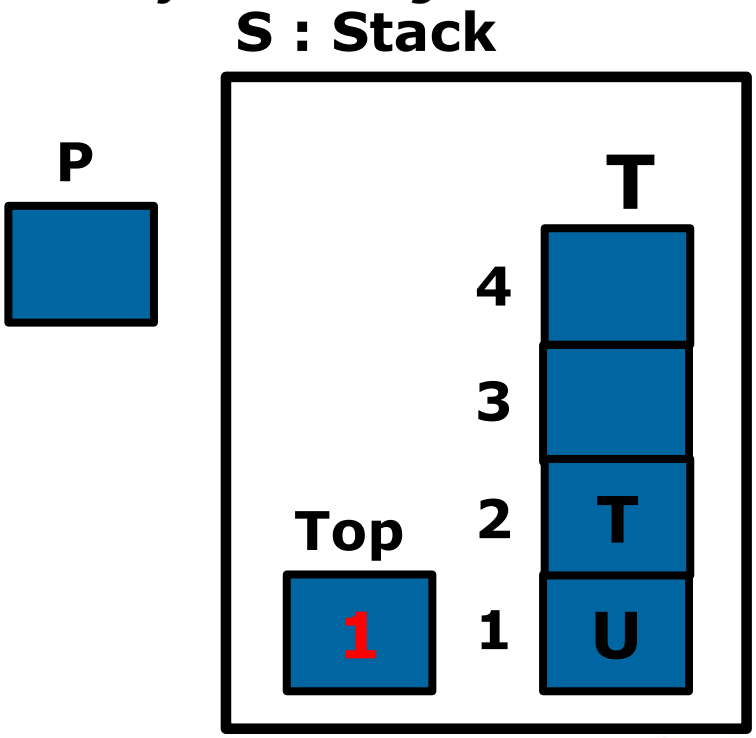
```
1.  procedure  Pop (Input/Output S : Stack, Output P : infotype)
2.  {I.S. Terdefinisi stack S mungkin kosong/penuh
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong.
4.  IS=FS jika stack kosong}
5.  kamus lokal
6.  algoritma
7.     if  isEmpty(S)  then
8.        output('Stack kosong')
9.     else
10.       P ← S.T[Top]
11.       S.Top ← S.Top – 1
12.    {end if}
13. {end procedure}
```



{Initial State}

PROSEDUR POP

```
1.  procedure  Pop (Input/Output S : Stack, Output P : infotype)
2.  {I.S. Terdefinisi stack S mungkin kosong/penuh
3.  F.S. Top S disimpan di P, stack mungkin menjadi kosong.
4.  IS=FS jika stack kosong}
5.  kamus lokal
6.  algoritma
7.     if  isEmpty(S)  then
8.        output('Stack kosong')
9.     else
10.       P ← S.T[Top]
11.       S.Top ← S.Top – 1
12.    {end if}
13. {end procedure}
```



{Final State}

TERIMA KASIH