□□□□□□□3

□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□/□ 4-35, □□□□

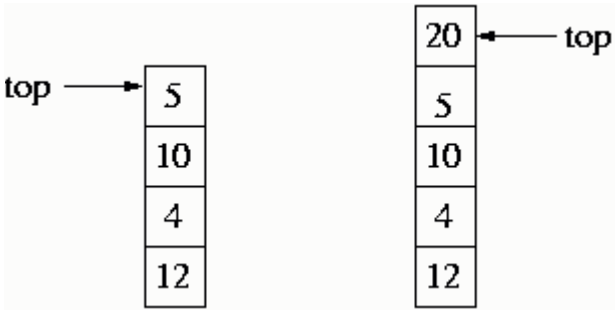
□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□



info *Pop(pointer S)*

```
if (S->Length == 0) return error
else
    x = Top(S);
    S->Length = S->Length –1;
    return x;
```

void *Push(info x, pointer S)*

```
if (S->Length == N) then error
else
    S->Length = S->Length + 1;
    S->Infos[S->Length-1] = x;
```

□□□□□□□□□□□□□□□□□□□□□□□□

Pop(): □(1)

Push(): □(1)

□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□/□ 4-35, □□□□

□□□□□□□□□□

Q: □□□, □□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

2□ □□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□/□ 4-35, □□□□

□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□2□□

□□□□□□□□□□□□□□□

□□□□□□Zig-Zag

□□□□□□□□□□

□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□