We assume that initial PDA was normalized to one with a single final state and that each step is either a PUSH (push new symbol on top of current top of stack) or POP (remove current top of stack). For the method from Hopcroft and Ullman, we start with a $ on bottom of stack and accept by final state and empty stack, For Sipser, we start with empty stack and accept by final state and empty stack. Also, Sipser's method requires read of element from Σ_e, whereas HU allows reading Σ_e^*.

Consider the pushdown automaton A = **({ q, f } , { 0 , 1 , c } , { 0, 1, c, $ }, δ, q, $, { f })**, where δ defines transitions:

δ(q , 0 , $) = { (q , PUSH(1)) }
δ(q , 1 , $) = { (q , PUSH(0)) }
δ(q , 0 , 0) = { (q , PUSH(1)) }
δ(q , 0 , 1) = { (q , PUSH(1)) }
δ(q , 1 , 0) = { (q , PUSH(0)) }
δ(q , 1 , 1) = { (q , PUSH(0)) }
δ(q , c , 0) = { (p , PUSH(c)) }
δ(q , c , 1) = { (p , PUSH(c)) }
δ(p , λ , c) = { (p , POP) }
δ(p , 0 , 0) = { (p , POP) }
δ(p , 1 , 1) = { (p , POP) }
δ(p , λ , $) = { (f , POP) }

This generates the language E(A) = **{ w c h(w)^R | w ∈ {0,1}$^+$ } and h(0)=1; h(1)=0**

Write the equivalent grammar using our class's variant of the construction in Hopcroft, Motwani and Ullman.
Hint: the starting non-terminal is: **< q , $, f >**, meaning generate all string that are consumed when we start in **q**, and end up in **f**, having uncovered what's below **$**.

< q, $, f > → 0 < q, 1, q > < q, $, f > **X**
| 0 < q, 1, p > < p, $, f >
| 0 < q, 1, f > < f, $, f > **X**
| 1 < q, 0, q> < q, $, f > **X**
| 1 < q, 0, p > < p, $, f >
| 1 < q, 0, f > < f, $, f > **X**
< q, 0, p > → 0 < q, 1, p > < p, 0, p >
| 1 < q, 0, p > < p, 0, p >
| c <p, 0, p >
< q, 1, p > → 0 < q, 1, p > < p, 1, p >
| 1 < q, 0, p > < p, 1, p >
| c <p, 1, p >
< p, 0, p > → 0
< p, 1, p > → 1
< p, $, f > → λ

< q, 0, q>, < q, 1, q >, < f, $, f > can lead nowhere as states q and f never entered after popping the stack.

Rewrite as
S → 0 T | 1 U
T → 0 T 1 | 1 U 1 | c 1 **// owe you a 1**
U → 0 T 0 | 1 U 0 | c 0 **// owe you a 0**

Consider the pushdown automaton A = **({ q, f } , { 0 , 1 , c } , { 0, 1, c }, δ, q, Φ, { f })**, where δ defines transitions:

δ(q , 0 , λ) = { (q , PUSH(1)) }
δ(q , 1 , λ) = { (q , PUSH(0)) }
δ(q , c , λ) = { (p , PUSH(c)) }
δ(p , λ , c) = { (p , POP) }
δ(p , 0 , 0) = { (p , POP), (f, POP) }
δ(p , 1 , 1) = { (p , POP), (f, POP) }

This generates the language $E(A)$ = **{ w c h(w)^R | w ∈ {0,1}$^+$ } and h(0)=1; h(1)=0**

Write the equivalent grammar using Sipser's construction.

The starting non-terminal is: **Aq,f**, meaning generate all string that are consumed when we start in **q**, and end up in **f** with the stack having the same contents as when we started in **q**. Note that the stack is empty at start and we allow top of stack to be ignored in transitions.

There are two cases for any **At,u**. Either something is pushed on stack and it gets back to its starting point at some intermediate state, **v**, and then back to the start at **u** **(At,u→ At,v Av,u)** or the first transition from **t** involves a **PUSH** and the last to **u** involves a **POP**. In this case, the input is of the form **xwy**, where the **x** is read at the **PUSH** and the **y** at the **POP**, where **x,y ∈Σ$_e$** and **w∈Σ$_e$***

Aq,f → Aq,q Aq,f	**Useless as becomes Aq,f→Aq,f since can never pop and end in q**
| Aq,p Ap,f	**Useless as can never push starting in p**
| Aq,f Af,f	**Useless as becomes Aq,f→Aq,f (see below) Note that: Aq,f Af,p is impossible**
| 0 Aq,p 1	**Would have pushed a 1 on stack when in q and must match it in p**
| 1 Aq,p 0	**Would have pushed a 0 on stack when in q and must match it in p**
Aq,p → Aq,q Aq,p	**Useless as becomes Aq,p→Aq,p since can never pop and end in q**
| Aq,p Ap,p	**Useless as becomes Aq,p→Aq,p (see below)**
| 0 Aq,p 1	**Would have pushed a 0 on stack when in q and must match it with 1 in p**
| 1 Aq,p 0	**Would have pushed a 1 on stack when in q and must match it with 0 in p**
| c Ap,p λ	**This reduces to Aq,p→c**
Aq,q → λ	**No other options as never pop and end in q**
Ap,p → λ	**No other options as never p pushes onto stack**
Af,f → λ	**No other options as never p pushes onto stack**

Rewrite as

S → 0 T 1 | 1 T 0
T → 0 T 1 | 1 T 0 | c