1. Peiyun(Seed) Zeng; Jing Lu; Pu Chai
2. 27
3. Oh boy, this is such an interesting question. Let me draw a pic for u

Computing + 9 + 9 10

Before first recursion

Current\_index = 1

First\_ operand (pac stack 1) = ?

Second\_perand(pac stack1) =?

Main stack

Parse\_and\_compute stack 1

First operand =? Second operand = ?

Step into the first call of parse\_and\_compute inside the main stack, initialize first operand and second operand (no value put in) (might be initialized to none)

Before second recursion Current\_index = 2

Main stack

PAC stack 2 will shrink after recursion

First operand =? Second operand = ?

Parse\_and\_compute stack 1

PAC stack 2

Return 8 and assign to first operand in pac stack 1

Before third recursion and fourth recursion

Current\_index = 3

First\_ operand (pac stack 1) = 8

Second\_perand(pac stack1) =?

Main stack

First operand =? Second operand = ?

Parse\_and\_compute stack 1

First operand =? Second operand = ? (2’)

PAC stack 2’

PAC stack 3

Return 9 and assign to first operand in pac stack 2’

So after recursion, pack stack 3 will srhink

Before fifth recursion

Current\_index = 5

First\_ operand (pac stack 1) = 8

Second\_perand(pac stack1) =?

Main stack

First operand =? Second operand = ?

Parse\_and\_compute stack 1

First operand =9 Second operand = ? (2’)

PAC stack 2’

PAC stack 3’

Return 10 and assign to first operand in pac stack 2’

Final recursion

Pack stack 3’ shrinks and inside pack stack2’ we get

First\_operand =9

Second\_operand = 10

Then we return 19 to THE SECOND OPERAND of PAC STACK 1, pack stack 2’ shrinks.

Now we left with this pretty little picture

Current\_index = 5

First\_ operand (pac stack 1) = 8

Second\_perand(pac stack1) =19

Main stack

Parse\_and\_compute stack 1

First operand =8 Second operand = 19

Now PAC stack 1 finally gets to shrink and return 27 to the main stack. HAHAH interesting process and that is why recursion is beautifully confusing.

4.

**Input: + 9 10**

**Output: 19**

**Input: + 8 + 9**

**error: caught exception integer 2**

5. ok. Another interesting question.

Step 1: first invoke PAC stack 1

Current\_index = 1, no error caught. Step into the if statement since encountered an “+”

Trying to calculate operand 1, update current\_index to 2 and invoke PAC stack 2 (inside PAC stack 1)

Step 2:

Now since 8 is not “+”, we go pass the if statement and return 8 directly to operand 1 (in PAC stack1). So PAC stack 2 shrinks and update current\_index to 3

Step 3:

Now inside PAC stack 1, we are trying to calculate operand 2. We invoke PAC stack 2’ (current\_index =3)

Since argv[3] = “+” we step into the if statement (recursion region). So inside PAC stack 2’, we update current\_index to 4 and invoke PAC stack 3.

Step 4:

Inside PAC stack3, since argv[4] = 9 so we pass the if statement (recursion region) and return 9 directly to first operand of PAC stack 2’ . PAC stack 3 shrinks [this step is very similar to step 2, only difference is now we are in the scope of stack 2’]

Step 5:

Now inside PAC stack 3, we trying to calculate second operand of PAC stack 2’ . we update current\_index to 5 and invoke PAC stack 3’

In the if statement of checking error, current\_index > last\_index (=4). We get error thrown. This command fails.

Ah interesting !

6. Two interesting errors. Another very intriguing question. Man I like this lab

(1) Start with current\_index = 2

This will mess things up a lot. The program no longer works as we expected

Take + 8 9 as an example. Now after analyzing through the debugger, we realize that since current\_index = 2 initially, we are gonna return 8 to the value directly

Another situation might be an invalid string will run through the program and get a value (this is really bad)

+ + 8 9

This should be invalid but the program will now get 17 since it basically starts calculating from argv[2]

(2) pass by copy vs pass by ref: current\_index vs & current\_index

This is another fatal error can be easily made. If we pass by copy, basically current\_index will not update.

No matter what we do, the program will basically return a value that is argv[2] + argv[2] if argv[1] is a “+”

For example + 9 8

The program will simply get 9 + 9 = 18