The git hub URL:

https://github.com/SaifAlnuaimi/CSC310\_Project1

1. Radix Sort

The radix sort is sorting the numbers in list by digit to digit starting from least significant digit to most significant digit. We have 10 digits for all. Then each number in list, we can put into one of 10 types. Then, we sort it by value of digit for the list of number.

Example: 11, 15, 12, 10. We do for first digit of each number, then we have: 1, 5, 2, 0 🡪 sorting by value of digit: 0, 1, 2, 5. So the list will sort as: 10, 11, 12, 15. Now continues set the next digit: 1, 1, 1, 1. It is equal so no change for order of list. Until now, the list is sorted.

To implement this algorithm in code with queue. We need do these steps below:

Step1: Loop each digit from least significant digit to most significant digit

Step2: For each digit, we need 10 queues respective [0-9] to put the number which have value of digit is [0-9].

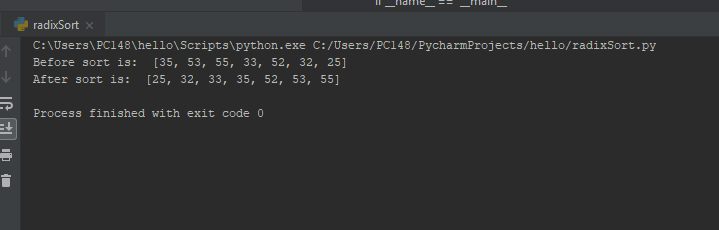
Step3: Check if this digit is last one to stop loop

Step4: Sorting the original list same with order of this digit. To do that, just initial list with number will get from queue[0] to queue[9].

Step5: repeat step2 to step4 for next digit until the last digit did.

Step6: return the list after sorted

Here is screenshot of program when running for sorting list: [35, 53, 55, 33, 52, 32, 25]



1. Postfix notation

As we know about the Postfix natation, it is an unambiguous way of writing an arithmetic expression

without parentheses. To evaluation the expression, we will go through the expression and check each character in it to see when the operator appears and then do the calculate. To do that, we need stack S. Each time when go thought the expression, we need push the number in to stack. And when see a operator in expression, we will pop two elements in stack for calculate the result and then push back this result into stack to use for next operator. We have problem with the number greater than 9 in expression (the number has more than 1 digit). Then we can use () to specify it. And with this assume, we need get correct number in code.

Here is detail of algorithm for Postfix evaluation “evalPostFix”:

Step1: loop each character in expression

Step2: Check if this is digit, ‘(‘, ‘)’ or operator

Step3: If this is digit, then add this character to number. If this is number has only one digit, then push it into stack

Step4: If this is ‘(‘, enable the flag to mark that this is number more than 1 digit

Step5: If this is ‘)’, disable the flag in step4 and put this number into stack

Step6: If this is operator, do these things below:

* Pop stack and save it as first number
* Pop stack and save it as second number
* Check if this operator is ‘+’, ‘-‘, ‘\*’ or ‘\’ and then get the results
* Push the result into stack

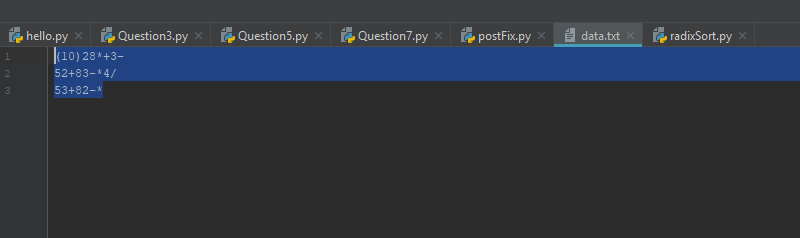
Step7: repeat the step2 to step6 until the end of expression

Step8: return the last result in stack by using pop.

To test this program, we need a list of expression in postfix notation. List of expression will store in a file input name “data.txt”. Each line is an expression.

Then the main program test will read each line in file and call the function evalPostFix to get the value for this expression.

The below is sample output when file data.txt has three expressions as below:



The screenshot of program:

