COMPSCI 119
LAB #2 – Bar Graphs

The goal of this Python programming assignment is to successfully create, enter, debug, and run a simple program in the Idle environment, based on a given framework containing a main program and the stubs of four functions. You are to complete the code inside those four functions.

Background

In this assignment you are going to write a program to enter a set of grades for students who have taken an exam, and then generate a set of text-based bar graphs based on those grades. For example, if student Fred received a 78 on his exam, the line of output for Fred will be his name, his score, his letter grade, and then 78 copies of the | character (the vertical bar), padded appropriately with blanks. When your program runs, the expected output from Idle/Python will look like the following image (the top image is the whole output, the bottom is the left part zoomed up so you can see it more easily):

```python
>>> Main()
Fred 78 C | | | | | | | | | | | | | | | | | | |
Sam 23 F | | | | | | | | | | | | | | | | | | |
Mary 100 A | | | | | | | | | | | | | | | | | | |
Bob 72 C | | | | | | | | | | | | | | | | | | |
Carol 62 D | | | | | | | | | | | | | | | | | | |
Sue 85 B | | | | | | | | | | | | | | | | | | |
Mortimer 5 F | | | | | | | | | | | | | | | | | | |
>>> }
```
**Setting up the Assignment**

Type in the following program code exactly as you see it here, with the same capitalization, except with your name instead of mine and the date included in the initial comment, and then save it in your Python folder with Lab2.py as the filename. Leave the gray areas blank for now; you will write new code there later.

```
# William T. Verts - Lab #2

def PadRight (S, N):
    Write new code here to return string S padded on right with spaces to length N.
    return S

def PadLeft (S, N):
    Write new code here to return string S padded on left with spaces to length N.
    return S

def GetGrade (Score):
    Write new code here to compute into variable Result a letter grade string appropriate to the value of variable Score.
    return Result

def Process (Name, Score):
    Write new code here to build a string S containing the name, score, letter grade, and histogram, all separated by the right number of spaces.
    return S

def Main ():
    print (Process ("Fred", 78))
    print (Process ("Sam", 23))
    print (Process ("Mary", 100))
    print (Process ("Bob", 72))
    print (Process ("Carol", 62))
    print (Process ("Sue", 85))
    print (Process ("Mortimer", 5))
    return
```

Notice that there are four functions which need to be finished, PadRight, PadLeft, GetGrade, and Process. You can complete and debug each function in order, before working on the next function in the program. Function Main is finished as-is and must not be modified from what you see here.
Task #1 – PadLeft and PadRight

The PadRight function has two parameters: \( S \) (a string) and \( N \) (an int), and returns as its result the string \( S \) padded on the right with blanks until the length of \( S \) contains no fewer than \( N \) characters. The PadLeft function is identical to PadRight except that it adds blanks to the left side of the string.

For example, the string "Frog" is four characters long, so PadLeft("Frog", 7) would return the seven-character string "     Frog" (with three extra blanks on the left) and PadRight("Frog", 7) would return the seven-character string "Frog     " (with three extra blanks on the right).

However, the string "Froggies" is already longer than seven characters, so both PadLeft("Froggies", 7) and PadRight("Froggies", 7) would return the original string "Froggies" unchanged as the result.

You must use a while loop in each function. You will also need to use the standard Python `len` function in your code to ask about the current length of a string, you will need string concatenation, but you may not use any other functions. Even so, there are at least two distinct approaches that can be used here that satisfy the requirements.

You are not allowed to use a for-loop or any Python function that pads strings.

You are not allowed to use the `+=` operator (we haven’t covered that yet).

You are not allowed to use string multiplication(such as "  "*X).

Complete these two functions.

Test these functions from the `>>>` command line prompt with different strings and different lengths to make sure that they handle all possible cases. That is, run the PadLeft function by typing `PadLeft("Frog", 7)` at the `>>>` prompt to see if you get "     Frog", for example, and then try the same task with PadRight. Do not proceed until these functions work correctly. Common problems include “off-by-one” errors, where the number of blanks might be one too few or one too many. Check this!

Note that PadLeft and PadRight return a value but do not print anything! You will lose points if either of these functions use the print statement! All printing happens in the Main function.
**Task #2 - GetGrade**

The GetGrade function has one int parameter called Score. The function must return the one-character string 'A' if Score is greater than or equal to 90, 'B' if Score is greater than or equal to 80, 'C' if Score is greater than or equal to 70, 'D' if Score is greater than or equal to 60, and 'F' otherwise.

You should also notice that the GetGrade function ends with the return Result statement. That means you must assign variable Result to have the correct letter grade value before exiting the function (not some other variable), and you may not use any other return statements in this function.

Complete the GetGrade function.

**Test the function** with different values for Score to make sure it handles all five grade ranges correctly. That is, test the function by typing GetGrade(90) at the >>> command prompt and see if you get 'A' as the result, but also test other values to make sure that the correct grade is returned for each value. Remember to test edge cases such as 89 and 90, 79 and 80, 69 and 70, and 59 and 60.

Note that GetGrade returns a value but does not print anything! You will lose points if this function uses the print statement! All printing happens in the Main function.
Task #3 - Process

The Process function has two parameters, Name and Score. Those parameters will be passed the name and the score for a particular student. For example, if the function is called as Process("Fred", 78) then Name will contain the string "Fred" and Score will contain the integer 78.

The Process function initializes a string S to the empty string, and then builds up and returns S as the value of the function, where S will contain the following items, in this order:

1: Student’s name, padded on the right to 10 characters (columns 1...10),
2: Student’s score, converted to a string, and padded on the left to 3 characters (columns 11...13),
3: Student’s letter grade, padded on the left to 2 characters (columns 14...15),
4: one extra blank (column 16),
5: the correct number of vertical bar characters (columns 17...17+Score-1).

This function starts to build the string S by calling PadLeft, PadRight, and GetGrade with the appropriate actual parameters.

This function then also requires the use of a loop to add to the string the correct number of vertical bar characters; for this assignment I want you to use a counter loop (a variable with a while-loop). You are not allowed to use a for-loop or any special Python functions or code that creates copies of strings (that is, you are not allowed to use any variant of the statement "|"*Score in your code).

Complete the Process function by writing code in the gray area to do all of this.

Note that Process returns a value but does not print anything! You will lose points if this function uses the print statement! All printing happens in the Main function.

Test the function by calling Process from the >>> command line prompt as Process("Fred", 78), which would return as the value of S the following string:

Fred       78 C |

Similarly, manually test Process("Sam", 23) and Process("Mary", 100) to verify that your function works correctly. Common problems include “off-by-one” errors, where the number of vertical bars might be one too few or one too many. Check this!
Running the Program

Run the program by typing `Main()` at the command prompt.

You should then get all seven lines of output as shown here, with the spacing as indicated in the following image:

```
>>> Main()
Fred  78 C
Sam   23 F
Mary  100 A
Bob   72 C
Carol 62 D
Sue   85 B
Mortimer 5 F
```

If the output does not contain the correct information, debug your program and run it again. Continue to edit and test your program until the resulting output does contain the correct information.

Finishing Up

When you are finished and everything runs correctly, submit the assignment for grading as Lab #2.

The graders will score your program by running `Main` to see if it correctly generates the appropriate graphs. You will also be graded on efficiency, completeness, and how well you follow the directions. The graders will be also testing each of your functions explicitly.
**Grading**

Study carefully the “Grading Codes” document on the class Web site. That is the generic rubric for all programming assignments in this class. It lays out the rules you must follow for all programs (you must have your name, lab number, and date in a comment at the top of the program, the program must run to completion without errors, it must solve the assigned problem, etc.) Each infraction gets a certain number of points removed and a letter code indicating which infraction was involved.

The generic rubric covers codes A-G and Z. However, here are two cases where the generic rubric needs extra explanation:

**F. -5  Program uses advanced methods not covered in class.** In this program, this code letter covers cases such as `PadLeft` and/or `PadRight` using functions from the Python string library, a `for`-loop, or the `+=` operator, or `Process` using a `for`-loop, for example. There are a lot of functions that can do the padding task for you, but I want students to learn the basics first. Similarly, the `for`-loop, multiplying a string by a constant, and the `+=` operator can all make the code tighter than it would be otherwise, but you have to learn to do things the hard way before taking any shortcuts. Walk before you run, and follow the directions!

**G. -5  Program works correctly but changes the assignment in order to do it.** In this program, all functions except `Main` return a value in a specific variable at the end of the function body. If any function returns its value internally through an additional `return` statement, the assignment has been changed and this penalty is incurred. If the template on page 2 is altered in any way, this penalty is incurred.

For this particular assignment, the additional error codes are as follows:

**H. -1  Missing lines of output.** There are seven students, so there should be seven lines of output. This error code will be assessed if there are more than or fewer than the specified seven lines.

**I. -1  Functions other than Main print their result.** Only `Main` is allowed to print; all other functions return a value.

**J. -1  Functions return the wrong data type.** A function that does not explicitly return a value can be said to return the special Python value `None`, which would affect the other functions that call it. Functions must return values of the correct type.

**K. -1  Functions return the wrong values.** `PadLeft` and `PadRight` might return the wrong number of characters (such as off-by-one errors), or `GetGrade` returns `'B'` for a Score of 90 (likely using `>` instead of `>=` in a comparison), or `Process` returns the wrong number of " | ".

**L. -1  The output is missing terms.** An example would be that the letter grade or score are missing, but everything else is there as required.

**M. -1  The output spacing is wrong.** Ignore this code if codes K or L occur. However, the output may still have incorrect spacing even when codes K or L do not occur (such as omitting the blank in column 16).