Suffix rules in Makefile

```
%.o: %.cpp %.h
    g++ -c –o $@ $<
```

- a rule that applies to all files ending in the .o suffix. The rule says that the .o file depends upon the .cpp version of the file and the .h files.
- $@ says to put the output of the compilation in the file named on the left side of the :, the $< is the first item in the dependencies list.

```
OBJ = main.o yourClass.o
a.out: $(OBJ)
    g++ -o $@ $^
```

- the special macros $@ and $^, which are the left and right sides of the :, respectively.

Useful UNIX Commands

Command	Description
man	help menu
pico	simple text editor
gcc	compiles your source code "gnu C compiler"
a.out	executes your program
ls –al	displays a long list of files "includes hidden files i.e. dot files"
pwd	prints working directory "pathname"
cd	changes directory
mkdir	creates a directory
rmdir	removes a directory
cp file1 file2	copies contents of file1 into file2
mv file1 file2	moves a file from one place to another, or change its name
rm	removes a file
more	displays a file's contents
grep	searches for a specified pattern in a file or list of files
ps	obtains the status of the active processes in the system
kill –9 pid	terminates all processes
passwd	modify a user's password
logout	terminates your session
who	display who is on the system
finger	displays the user information
date > myfile	"output redirection" saves the output of date command in myfile
cal >> myfile	"appends" calendar to myfile
cal	display a calendar and the date
wc file1	counts the number of lines, words, and characters in file1

This page intentionally left blank.

Homework #1 Postfix Expressions

- Write a program to read an postfix expression from the keyboard, evaluate its value, and print the result.
- For simplicity you can assume that:

1. The postfix expression is valid.
2. Only binary operators are used: + (addition), - (subtraction), * (multiplication), / (division), % (modulus) and ^ (exponentiation).
3. The operands are positive integers between 1 and 9, inclusive, separated by one or more blank space characters.

- Test your program with at least 10 different input expressions before submitting your solution.

HW #1 (2)

- Sample run: This is a typical example of how your program should work (in interactive mode):

Enter the postfix expression: 7 8 3 * - 6 2 ^ + 3 – 7 2 / +

The value of the expression is: 19

- The program should then ask the user if another expression should be performed. Accept either "y" or "Y" as positive responses.

HW #1 (3)

- Your task is to:
  1. Implement the program using C++. **[或是像 C 的 C++]**
  2. Download Java J2SE from
     http://java.sun.com/j2se/
     and rewrite the program in Java.
  3. Download Python 3 from
     https://www.python.org/downloads/
     and rewrite the program in Python.

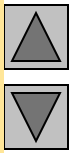
- You are required to submit a single "**makefile**" as well.

HW #1 (4)

- Learn how to set up your **debugging** environment and get experience in, for example, setting breakpoints, stepping through the execution, and evaluating a variable or an expression.

```
1  // Fig. 15.1: fig15_01.cpp
2  // Addition program
3  #include <iostream>
4
5  int main()
6  {
7     int integer1, integer2, sum;           // declaration
8
9     std::cout << "Enter first integer\n";  // prompt
10    std::cin >> integer1;                  // read an integer
11    std::cout << "Enter second integer\n"; // prompt
12    std::cin >> integer2;                  // read an integer
13    sum = integer1 + integer2;             // assignment of sum
14    std::cout << "Sum is " << sum << std::endl; // print sum
15
16    return 0;   // indicate that program ended successfully
17 }
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```



Outline

1. Load <iostream>

2. main

2.1 Initialize variables integer1, integer2, and sum

2.2 Print "Enter first integer"

2.2.1 Get input

2.3 Print "Enter second integer"

2.3.1 Get input

2.4 Add variables and put result into sum

2.5 Print "Sum is"

2.5.1 Output sum

2.6 exit (return 0)

Program Output

- The following is designed to familiarize you with the mechanics of creating, editing, compiling, and running a text-mode Java application.
- You do not have to hand it in, but you should write and run it.
- The source code in the following pages simply prompts for and accepts two numbers from the user, adds them, and displays the result.
- The file name, *Add.java* is case-sensitive and must match the class name in the program.

```
/*************************************************************
    Program to add two numbers... note that input is accepted as a
     String and then an attempt is made to convert it to a integer for
     calculations. Non-numeric input is detected by the Exception
     mechanism and a default value is assigned to the value.
     *************************************************************/
import java.io.*;
import java.util.Scanner;

public class Add {
    public static void main(String args[]) {
        String amtStr;
        int num1 = 0, num2 = 0, tot = 0;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the first number: ");
        amtStr = sc.next();
        // try to convert amt String to integer for calculation
        try {  num1 = new Integer(amtStr).intValue();  }
        catch (NumberFormatException  e) {
            System.out.println("Bad numeric input; 1st num set to 100");
            num1 = 100; }
```

```
        System.out.println("Enter the second number: ");
        amtStr = sc.next();

        try {  num2 = new Integer(amtStr).intValue();  }
        catch (NumberFormatException  e) {
           System.out.println("Bad numeric input; 2nd num is set to 50");
           num2 = 50; }

        tot = num1 + num2;

        System.out.println("Sum is: " + tot);

    } // end main
}   // end of class Add
```

HW #1 (5)

- The **Python** program below calculates the sum of two numbers entered by the user.

```
# Store input numbers
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')

# Add two numbers
sum = float(num1) + float(num2)

# The zero argument is the name of the program file when using command line
# sum = float (sys.argv[1]) + float (sys.argv[2])

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

HW #1 (6)

- Output:

Enter first number: 1.5

Enter second number: 6.3

The sum of 1.5 and 6.3 is 7.8

- We use the built-in function *input()* to take the input. Since, *input()* returns a string, we convert the string into number using the *float()* function. Then, the numbers are added.

The Python Standard Library

- https://docs.python.org/2/library/

- **Hint:** use the **math.sqrt** function to compute the square root. (If you are using the Python interpreter, you need to first do **import math**)

Classes: A First Look

```
#include <iostream.h>

#define SIZE 10

// Declare a stack class for characters
class stack {
    char stck[SIZE]; // holds the stack
    int tos;              // index of top-of-stack

public:
    void init();              // initialize stack
    void push(char ch); // push character on stack
    char pop();               // pop character from stack
}
```

```
// Initialize the stack
void stack::init() { tos = 0; }

// Push a character.
void stack::push(char ch) {
    if (tos==SIZE) { cout << “Stack if full”; return; }
    stck[tos] = ch;
    tos++; }

// Pop a character
char stack::pop() {
    if (tos==0) { cout << “Stack is empty”;
                  return 0; // return null on empty stack
                }
    tos--; return stck[tos]; }
```

```
main() {
    stack s1, s2; // create two stacks
    int i;
    // initialize the stacks
    s1.init();
    s2.init();

    s1.push(‘a);          s2.push(‘x’);
    s1.push(‘b’);         s2.push(‘y’);
    s1.push(‘c’);         s2.push(‘z’);

    for (i=0; i<3; i++) cout << “Pop s1: “ << s1.pop() << “\n”;
    for (i=0; i<3; i++) cout << “Pop s2: “ << s2.pop() << “\n”;

    return 0;
}
```