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.

Homework #1:

Characters Frequency Calculator

- In this assignment you are asked to write a program that will take-in multiple command line arguments and output statistical information about these arguments.

- More specifically, your program will output a table listing all the characters (letters, numbers, spaces, or whatever .. i.e., you do not have to check if the contents of the passed-in string are alphabet letters).

HW #1 (2)

- Here are some example runs:

<C++> ./Charfreq “HELLO CLASS”

<Java> Java Charfreq “HELLO CLASS”

H-1

E-1

L-3

O-1

-1

C-1

A-1

S-2

HW #1 (3)

<C++> ./Charfreq i can count 123

<Java> Java Charfreq i can count 123

i-1

c-2

a-1

n-2

o-1

u-1

t-1

1-1

2-1

3-1

HW #1 (4)

- Notice that each character is printed in the table exactly once.
- Also notice that the space character was printed in the first example but not the second because the first example had only one command line argument which happened to contain the space (that is why the quotes were used), while in the second example, three separate command line arguments were passed-in and no spaces were in any of them.

HW #1 (5)

- The characters must be printed in the order as they appear in the arguments list.

- Your task is to:
  1. Implement the program using C++.
  2. Download Java J2SE from
     http://java.sun.com/j2se/
     and rewrite the program in Java.

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

```
// Here we are asked to read two numbers in command line and
// display the sum.
// One sample run test: a.out 3 4
// This program shows you the basics of building a C++ program.
//
#include <iostream>
// Using statements which allow you to use the streams/keywords
// in the std:: namespace without needing the std:: prefix
using std::cout;
using std::endl;

// Solution implemented entirely in main()
int main (int argc, char *argv[]) {
  if (argc != 3) {
   cout << "You have forgot to specify two numbers." << endl; exit(1); }
  cout << "The sum is : " <<  atoi (argv[1]) + atoi (argv[2]);
  return 0;
}
```

HW #1 (6)

- 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.

- If you are stuck or are having trouble, you may examine the following C code.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
typedef struct _char_count {
  char c;
  int count;
  struct _char_count *next;
} char_count;

int main (int argc, char **argv) {
  int i, j;
  char_count *headptr, *tmp, *prev;

  tmp = NULL;
  prev = NULL;
  headptr = NULL;
```

我們起個頭， 剩下的部份請自己想辦法...

- 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 prints all command-line parameters on standard error.
- The file name, *EchoArgs.java* is case-sensitive and must match the class name in the program.

```
public class EchoArgs
{
  public static void main(String[] args)
  {
    System.err.println("EchoArgs called, args.length = " + args.length);
    for (int i = 0; i < args.length; i ++)
      System.err.println("EchoArgs args[" + i + "] = <" + args[i] + ">");

  } // end of main() method

} // end of EchoArgs class
```