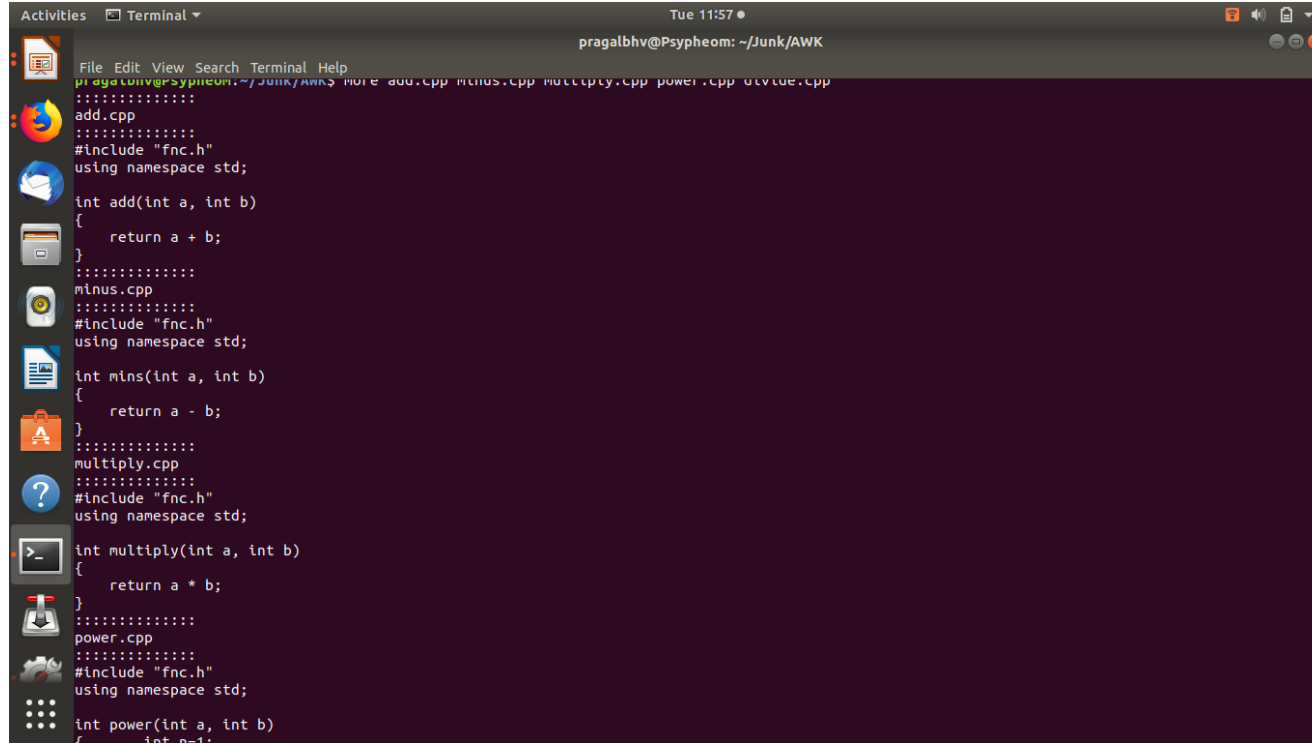


# HOMEWORK 5

Makefile

Take one of your old codes, split the code into separate files, one for each function. Create a makefile and test the recompilation



The screenshot shows a terminal window titled "pragalbhv@Pyspheom: ~/Junk/AWK" with a timestamp of "Tue 11:57". The terminal displays the contents of several C++ files, including `add.cpp`, `minus.cpp`, `multiply.cpp`, `power.cpp`, and `divide.cpp`. Each file contains a function definition that includes `"fnc.h"` and uses the `std` namespace. The functions are `add`, `mins`, `multiply`, and `power`. The `add` function returns `a + b`, `mins` returns `a - b`, `multiply` returns `a * b`, and `power` returns `a * b`. The terminal also shows a list of files in the directory: `add.cpp`, `minus.cpp`, `multiply.cpp`, `power.cpp`, and `divide.cpp`.

```
pragalbhv@Pyspheom: ~/Junk/AWK
more add.cpp minus.cpp multiply.cpp power.cpp divide.cpp
add.cpp
#include "fnc.h"
using namespace std;

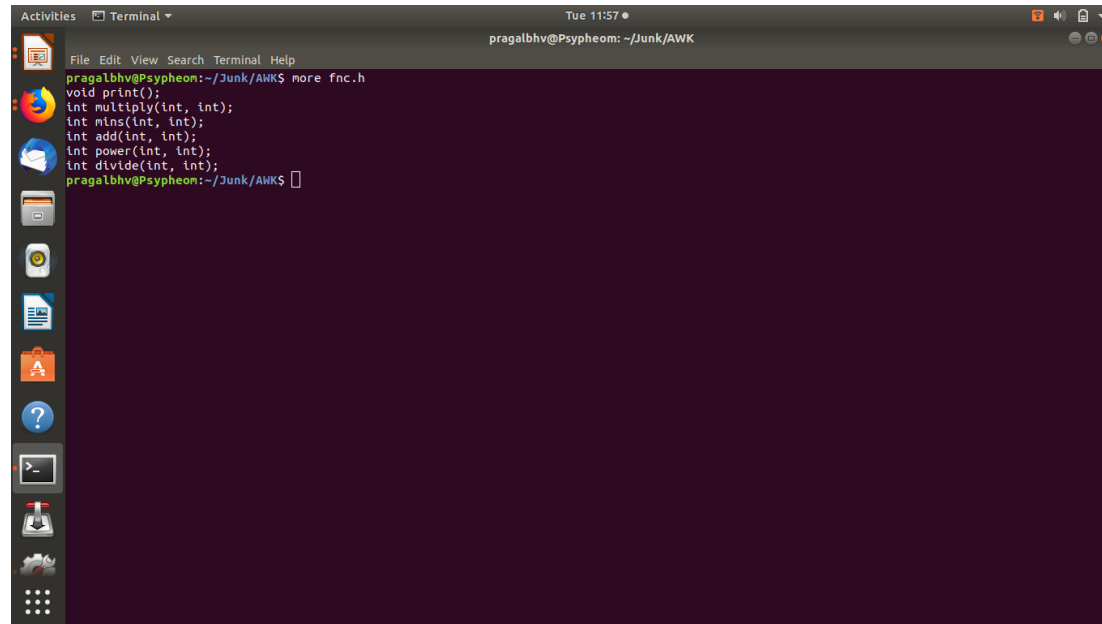
int add(int a, int b)
{
    return a + b;
}
minus.cpp
#include "fnc.h"
using namespace std;

int mins(int a, int b)
{
    return a - b;
}
multiply.cpp
#include "fnc.h"
using namespace std;

int multiply(int a, int b)
{
    return a * b;
}
power.cpp
#include "fnc.h"
using namespace std;

int power(int a, int b)
{
    int n=1;
```

# Fnc.h



A terminal window titled "pragalbhv@Pspheon: ~/Junk/AWK" is shown. The window has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal output shows the command `more fnc.h` being executed, displaying the contents of the file `fnc.h`. The file contains the following code:

```
void print();
int multiply(int, int);
int mins(int, int);
int add(int, int);
int power(int, int);
int divide(int, int);
```

The terminal prompt is `pragalbhv@Pspheon:~/Junk/AWK$`.

Activities Terminal

pragalbhv@Pspheon:~/Junk/AMK\$ more fnc.h

```
void print();
int multiply(int, int);
int mins(int, int);
int add(int, int);
int power(int, int);
int divide(int, int);
```

pragalbhv@Pspheon:~/Junk/AMK\$

Activities Terminal Tue 12:00

pragalbhv@Pspheon:~/Junk/AMK\$

File Edit View Search Terminal Help

```
pragalbhv@Pspheon:~/Junk/AMK$ make
g++ -g3 -ggdb -O main.o add.o minus.o multiply.o power.o divide.o -o main.e
pragalbhv@Pspheon:~/Junk/AMK$ ./main.e
ENTER TWO NOS.8
2
MULTIPLY16
MINUS6
ADD10
int DIVIDE4
POWER64
```

pragalbhv@Pspheon:~/Junk/AMK\$

File Edit View Search Terminal Help

pragalbhv@Pyspheom:~/Junk/AWK\$ more Makefile

CC = g++

CC = g++

CFLAGS = -g3 -ggdb -O

default: main.o add.o minus.o divide.o power.o multiply.o function.h

\$(cc) \$(CFLAGS) main.o add.o minus.o multiply.o power.o divide.o -o main.e

main.o: main.cpp fnc.h

\$(cc) \$(CFLAGS) -c main.cpp -o main.o

add.o: add.cpp fnc.h

\$(cc) \$(CFLAGS) -c add.cpp -o add.o

minus.o: minus.cpp fnc.h

\$(cc) \$(CFLAGS) -c minus.cpp -o minus.o

multiply.o: multiply.cpp fnc.h

\$(cc) \$(CFLAGS) -c multiply.cpp -o multiply.o

divide.o: divide.cpp fnc.h

\$(cc) \$(CFLAGS) -c divide.cpp -o divide.o

power.o: power.cpp fnc.h

\$(cc) \$(CFLAGS) -c power.cpp -o power.o

DATESTAMP=\$(shell date +"%Y-%m-%d")

TARBALL=codebackup\_\$(DATESTAMP).tar

tar:

# ----- making a tarball -----

@echo "Backing up with datestamp: \$(DATESTAMP)";

@echo "Tarball name: \$(TARBALL)";

tar -cvf \$(TARBALL) \*.cpp \*.hpp readme.txt Makefile

@ls -l \$(TARBALL)

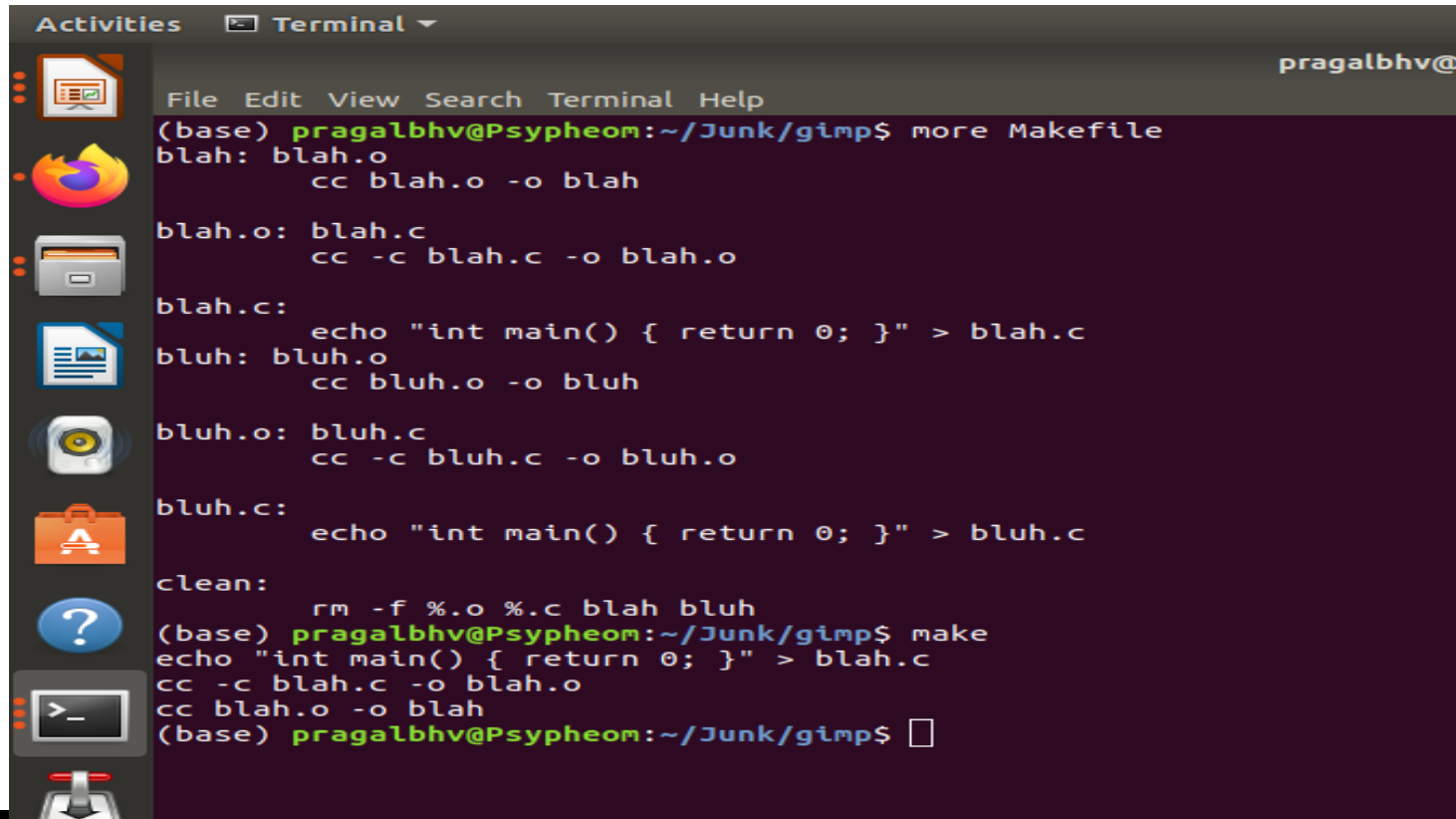
/bin/mv \$(TARBALL) \$(BACKUPDIR)

# ----- done moving tarball -----

# ----- end of Makefile -----

pragalbhv@Pyspheom:~/Junk/AWK\$

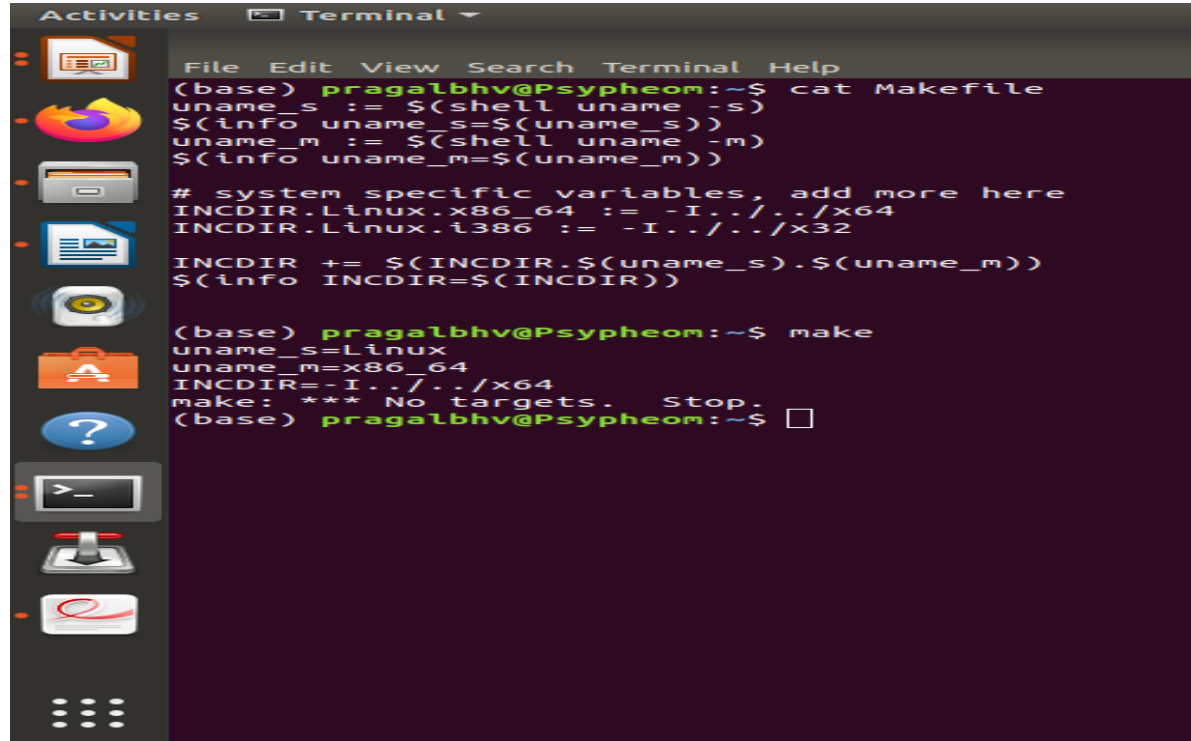
Create a makefile that uses a pattern for files rather than explicit listing of each of the files.



The image shows a terminal window with a dark background and a sidebar on the left containing icons for various applications. The terminal output is as follows:

```
Activities  Terminal ▾  
pragalbhv@  
File Edit View Search Terminal Help  
(base) pragalbhv@Psypheom:~/Junk/gimp$ more Makefile  
blah: blah.o  
      cc blah.o -o blah  
  
blah.o: blah.c  
      cc -c blah.c -o blah.o  
  
blah.c:  
      echo "int main() { return 0; }" > blah.c  
bluh: bluh.o  
      cc bluh.o -o bluh  
  
bluh.o: bluh.c  
      cc -c bluh.c -o bluh.o  
  
bluh.c:  
      echo "int main() { return 0; }" > bluh.c  
  
clean:  
      rm -f %.o %.c blah bluh  
(base) pragalbhv@Psypheom:~/Junk/gimp$ make  
echo "int main() { return 0; }" > blah.c  
cc -c blah.c -o blah.o  
cc blah.o -o blah  
(base) pragalbhv@Psypheom:~/Junk/gimp$
```

Prepare a Makefile that performs conditional compilation depending on the architecture of the machine.

A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Help) and a sidebar with application icons. The terminal shows the creation of a Makefile and its execution. The Makefile defines variables for system-specific paths based on the architecture (x86\_64 or i386). The execution shows the variables being set and the message '\*\*\* No targets. Stop.'

```
(base) pragalbhv@Psypheom:~$ cat Makefile
uname_s := $(shell uname -s)
$(info uname_s=$(uname_s))
uname_m := $(shell uname -m)
$(info uname_m=$(uname_m))

# system specific variables, add more here
INCDIR.Linux.x86_64 := -I.././x64
INCDIR.Linux.i386 := -I.././x32

INCDIR += $(INCDIR.$(uname_s).$(uname_m))
$(info INCDIR=$(INCDIR))

(base) pragalbhv@Psypheom:~$ make
uname_s=Linux
uname_m=x86_64
INCDIR=-I.././x64
make: *** No targets. Stop.
(base) pragalbhv@Psypheom:~$
```

# Create a Makefile that uses your own bash shell scripts in each recipe.

```
pragalbhv@Psyspheom: ~/Junk/makemake
File Edit View Search Terminal Help
(base) pragalbhv@Psyspheom:~/Junk/makemake$ more Makefile
default:
    @echo "File is Not Specified"
script1:
    @./script1.sh
script2:
    @./script2.sh

(base) pragalbhv@Psyspheom:~/Junk/makemake$ more script1.sh
#!/bin/bash
echo "enter value of a";
read a;
echo "enter value of b";
read b;

echo "$a+$b" | bc;

(base) pragalbhv@Psyspheom:~/Junk/makemake$ more script2.sh
#!/bin/bash
HELLO=Hello
function hello {
    local HELLO=World
    echo $HELLO
}

echo $HELLO
hello
echo $HELLO
(base) pragalbhv@Psyspheom:~/Junk/makemake$ make
File is Not Specified
(base) pragalbhv@Psyspheom:~/Junk/makemake$ make script1.sh
make: Nothing to be done for 'script1.sh'.
(base) pragalbhv@Psyspheom:~/Junk/makemake$ make script1
enter value of a
2
enter value of b
2
4
(base) pragalbhv@Psyspheom:~/Junk/makemake$ make script2
```



```
ies Terminal ▾ Mon 20:43 ●
pragalbhv@Psyspheom: ~/Junk/makemake

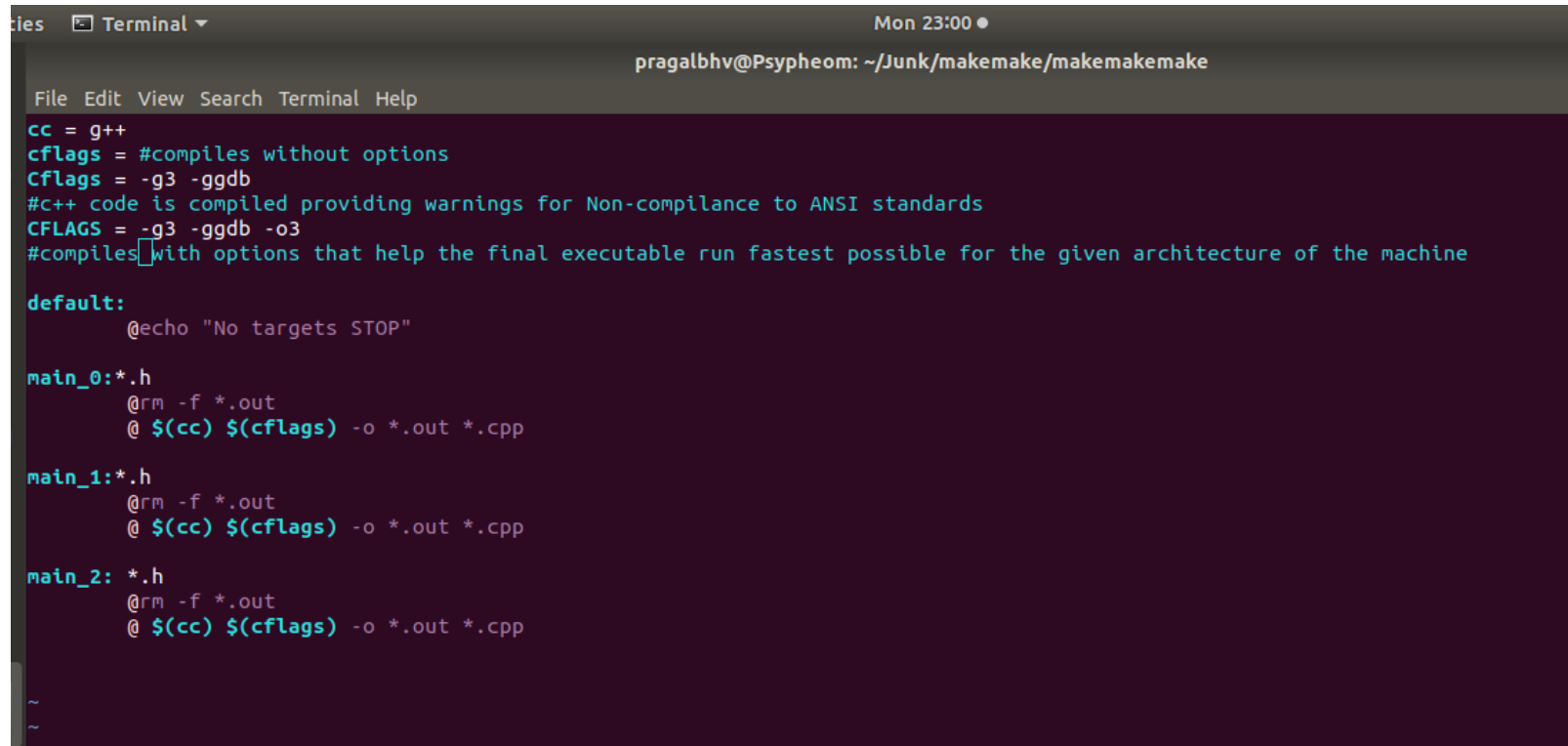
File Edit View Search Terminal Help
(base) pragalbhv@Psyspheom:~/Junk/makemake$ more Makefile
default:
    @echo "File is Not Specified"
script1:
    @./script1.sh
script2:
    @./script2.sh

(base) pragalbhv@Psyspheom:~/Junk/makemake$ more script1.sh
#!/bin/bash
echo "enter value of a";
read a;
echo "enter value of b";
read b;

echo "$a+$b" | bc;

(base) pragalbhv@Psyspheom:~/Junk/makemake$ more script2.sh
#!/bin/bash
HELLO=Hello
function hello {
    local HELLO=World
    echo $HELLO
}
echo $HELLO
hello
echo $HELLO
(base) pragalbhv@Psyspheom:~/Junk/makemake$ make
File is Not Specified
(base) pragalbhv@Psyspheom:~/Junk/makemake$ make script1.sh
make: Nothing to be done for 'script1.sh'.
(base) pragalbhv@Psyspheom:~/Junk/makemake$ make script1
enter value of a
2
enter value of b
2
4
(base) pragalbhv@Psyspheom:~/Junk/makemake$ make script2
```

Create a Makefile that can compile a code in three different ways namely (a) without any options, (b) with all options to provide warnings for non-compliance to ANSI standards etc., and (c) with options that help the final executable run fastest possible for the given architecture of the machine



```
pragalbhv@Psyspheom: ~/Junk/makemake/makemakemake
File Edit View Search Terminal Help
cc = g++
cflags = #compiles without options
Cflags = -g3 -ggdb
#c++ code is compiled providing warnings for Non-compliance to ANSI standards
CFLAGS = -g3 -ggdb -o3
#compiles with options that help the final executable run fastest possible for the given architecture of the machine

default:
    @echo "No targets STOP"

main_0:*.h
    @rm -f *.out
    @ $(cc) $(cflags) -o *.out *.cpp

main_1:*.h
    @rm -f *.out
    @ $(cc) $(Cflags) -o *.out *.cpp

main_2: *.h
    @rm -f *.out
    @ $(cc) $(CFLAGS) -o *.out *.cpp

~
~
```