1 Written Section

1. Describe at least 2 ways of transferring file from a remote sever to a local machine.

   (a) **FTP** or **Internet File Transfer Program** is an internet standard file transfer protocol. The program let user transfer file between local and remote machine.
   Synopsis: ftp [-46pinegvd] [host [port]]

   (b) **SFTP** or **Safe File Transfer Program** is pretty much the same as FTP, except it operates on encrypted ssh transport.
   Synopsis: sftp [-1246Cpqrv] [user@] host[:file]

2. What are revision control systems? Why are they useful?

   RCS or **VCS** or**Revision Control Systems** or **Vision Control** is a system that keep track of the changes that has done to a file. It allows you to revert files back to a previous version, review change made over time, see who last modified the file, etc.. There are many types of revision control system:

   (a) **Local Version Control** – the VCS keeps the history of file changes in a simple database on the local computer.

   (b) **centralized Version Control** – the VCS keeps the file changes in a central VCS, a sever, along with keeping track of who is checking out the file.

   (c) **Distributed Version Control** – the clients doesn't check out only the lastest snapshot of teh file: they fully mirror the repository. Thus, if the server dies, any of the client repositories can be opied back up to the sever to restore it.

3. What is the difference between redirecting and piping? Describe each.

   - **Redirecting** is like changing the output file of a program to be the input of another program. (Program A creates a single output file, then Program B reads from it)
   - **Piping** is changing the output flow of a program to be the input of another program. So instead of the output of the first program being the input of the second program, you are going to have the data of the output flows into the input of the second program. (Program A writes to the Pipe, and Program B then reads from the Pipe)

4. What is make, and how is it useful?
   make is a unix command that will search the current working directory for *makefile*, or *Makefile* (in order) and goes through a descriptor file starting from **target**, looking for **dependencies**, and chain of dependencies until it reaches the end of the chain.
   Synopsis: make [-f makefile] [-bBdeiknpqrsSt] [macro name=value] [names]

5. Describe, in detail, the syntax of a make file.
   Syntax: *target: dependencies*
   [tab]system command

   - **target** - a set of instrunctions on how to generates a file.

- **dependencies** - a baby *target*.
  Note it is very useful to use different targets because it allows you to compile a part that has been changed, rather than recompiling the whole program.
- **variable** - it is a variable for system command. (Conventionally, variables are named in all caps)
- **comment** - comment. Anything on the same line as # is a comment and will not be execute once the make is called.

Example:

```
CC=g++
CFLAGS=-c -Wall
all: prog

prog: main.o file1.o file2.o
$(CC) main.o file1.o file2.o -o prog

file1.o: file1.cpp
# this is a comment for this part in makefile
$(CC) $(CFLAGS) file1.cpp

file2.o: file2.cpp
$(CC) $(CFLAGS) file2.cpp

clean:          rm -rf *o prog
```

6. Give a *find* command that will run the *file* command on every regular file (NO DIRECTORIES) within the current filesystem subtree.
   From EXAMPLES section of *man find*, I think the closest anwer is
   find . -type f -exec file '{}' ;