Describe the pattern represented by the following regular expression, give an example of string that can match it:
"^\(?[2-9][0-9]{2}\)?(|-|\.)[0-9]{3}(|-|\.)[0-9]{4}"
This regex matches 10-digit phone numbers, with three kinds of delimiter: none, -, .
The first three digits(zone number) can be either with () or not. A zone number cannot start with 0 or 1.
ex: 2345678999, (234)5678999, 234-567-8999

1) How would you know the differnces between the the working copy of bar.c and bar.h, an their respective versions in a commit that was made just before the last commit (penultimate commit)? You cannot use commit hashes to refer to commits.
Ans: git status
git diff
Explain the steps of dynamic linking:
Dynamic linking defers much of the linking process until a program starts running. It provides a variety of benefits that are hard to get otherwise:
Dynamically linked shared libraries are easier to create than static linked shared libraries.
Dynamically linked shared libraries are easier to update than static linked shared libraries.
The semantics of dynamically linked shared libraries can be much closer to those of unshared libraries.
Dynamic linking permits a program to load and unload routines at runtine, a facility that can otherwise be very difficult to provide.
There are a few disadvantages, of course. The runtime performance costs of dynamic linking are substantial compared to those of static linking, since a large part of the linking process has to be redone every time a program runs. Every dynamically linked symbol used in a program has to be looked up in a symbol table and resolved. (Windows DLLs mitigate this cost somewhat, as we describe below.) Dynamic libraries are also larger than static libraries, since the dynamic ones have to include symbol tables.

--start a linker---
The linker then initializes a chain of symbol tables with pointers to the program's symbol table and the linker's own symbol table. Conceptually, the program file and all of the libraries loaded into a process share a single symbol table. But rather than build a merged symbol table at runtime, the linker keeps a linked list of the symbol tables in each file. each file contains a hash table to speed symbol lookup, with a set of hash headers and a hash chain for each header. The linker can search for a symbol quickly by computing the symbol's hash value once, then running through apprpriate hash chain in each of the symbol tables in the list.
--find libraries---
Once it's found the file containing the library, the dynamic linker opens the file, and reads the ELF header to find the program header which in turn points to the file's segments including the dynamic segment. The linker allocates space for the library's text and data segments and maps them in, along with zeroed pages for the bss. From the library's dynamic segment, it adds the library's symbol table to the chain of symbol tables, and if the library requires further libraries not already loaded, adds any new libraries to the list to be loaded.
When this process terminates, all of the libraries have been mapped in, and the loader has a logical global symbol table consisting of the union of all of the symbol tables of the program and the mapped library.
Lazy:
All calls within the program or library to a particular routine are adjusted when the program or library is built to be calls to the routine's entry in the PLT. The first time the program or library calls a routine, the PLT entry calls the runtime linker to resolve the actual address of the routine. After that, the PLT entry jumps directly to the actual address, so after the first call, the cost of using the PLT is a single extra indirect jump at a procedure call, and nothing at a return.

real 0m4.866s: elapsed time as read from a wall clock
user 0m0.001s: the CPU time used by your process
sys 0m0.021s: the CPU time used by the system on behalf of your process

```
--- lib/utimens.c~    2005-11-02 04:16:12.000000000 -0800
+++ lib/utimens.c    2010-01-26 12:20:13.292737899 -0800
@@ -71,7 +71,7 @@ struct utimbuf
   Return 0 on success, -1 (setting errno) on failure.  */
 int
-futimens (int fd ATTRIBUTE_UNUSED,
+coreutils_futimens (int fd ATTRIBUTE_UNUSED,
               char const *file, struct timespec const timespec[2])
 {
   /* There's currently no interface to set file timestamps with
@@ -160,5 +160,5 @@ futimens (int fd ATTRIBUTE_UNUSED,
 int
 utimens (char const *file, struct timespec const timespec[2])
 {
-  return futimens (-1, file, timespec);
+  return coreutils_futimens (-1, file, timespec);       }
```