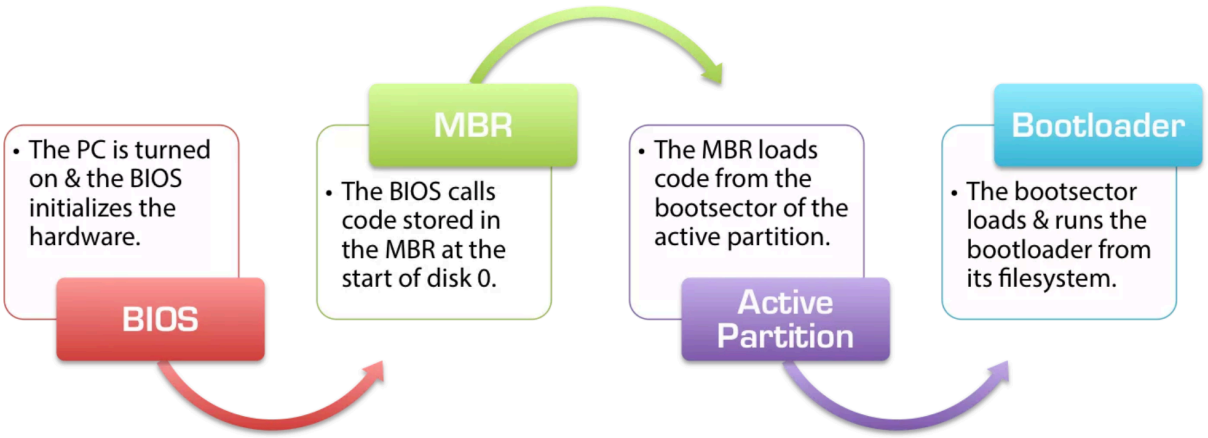
1.1 Problem 1 What is the purpose of the BIOS in the classical IBM-PC architecture? Explain the relationship between the BIOS, Master Boot Record, and Boot Loader (draw a picture).

BIOS (basic input/output system) is the program a personal computer's microprocessor uses to get the computer system started after you turn it on. It also manages data flow between the computer's operating system and attached devices such as the hard disk, video adapter, keyboard, mouse and printer.



1.2 Problem 2 What characteristics of the UEFI firmware standard make UEFI PCs more suspectible to the risk of rootkit attacks? How does the standard attempt to minimize this risk?

The standard UEFI codebase now includes a rich set of network capabilities for Ethernet, WiFi, and even Bluetooth that allow the firmware to communicate remotely and even perform a full HTTP boot from a remote server across the internet. What this can allow is a buffer overflow vulnerability and allow for arbitrary remote code execution.

The standard attempts to rectify this by verifying keys that are located within several components of the machine and validate that they are the expected keys as well as constantly keeping a back up of previous BIOS version so the BIOS can return itself to a healthy state in the event of an attack.

1.3 Problem 3 When using the Linux-specific reboot() system call to reboot the system, the second argument, magic2, must be specified as one of a set of magic numbers (e.g., LINUX_REBOOT_MAGIC2). What is the significance of these numbers? (Converting them to hexadecimal provides a clue.)

The significance of those numbers are the birth date of Linus Torvald and his 3 daughters.

1.4 Problem 4

```
vector<string> parseCommandLine(string aCommandLine) {
  vector<string> parsed;

  string build;

  for (int pos = 0; pos < aCommandLine.size(); ++pos) {
    if(aCommandLine[pos] == '|'){
      parsed.push_back(build);
      ++pos;
      build.clear();
    }
    else if(aCommandLine[pos] == '>'){
      parsed.push_back(build);
      ++pos;
      build.clear();
    }
    else if(aCommandLine[pos] == '<'){
      parsed.push_back(build);
      ++pos;
      build.clear();
    }
    else if(aCommandLine[pos] == '<' && aCommandLine[pos+1] == '<')
    {
      parsed.push_back(build);
      pos += 2;
      build.clear();
    }
    else {
      build += aCommandLine[pos];
    }
  }
  return parsed;
}
```