HOMEWORK 3

Q-1. Describe some of the challenges of designing operating systems for mobile devices compared with designing operating systems for traditional PCs.

Designing operating systems for mobile devices presents distinct challenges compared to traditional PCs. One major issue is **hardware limitations**: mobile devices have less processing power, memory, and storage, and are constrained by battery life. The OS must efficiently manage these resources to maintain performance and extend battery usage, often suspending background processes and optimizing I/O operations.

Another challenge is **mobility and connectivity**. Unlike PCs, mobile devices frequently switch between networks (Wi-Fi, LTE, 5G) and encounter variable signal strength. The OS must ensure seamless transitions and maintain active network sessions without disrupting user experience.

User interaction is also different. Mobile devices rely on touchscreens, voice commands, and sensors. The OS must provide low-latency, adaptive UI frameworks that support a wide variety of screen sizes and input methods.

Security concerns are heightened on mobile platforms. Devices are more likely to be lost or stolen and are constantly connected to networks. OS design must include strong app sandboxing, secure boot mechanisms, and fine-grained permission controls to protect user data.

Lastly, **hardware diversity and vendor customizations** require the OS to support a wide range of devices while maintaining performance and compatibility. Together, these factors make mobile OS design a complex, multidisciplinary task.

Q-2. Identify several advantages and several disadvantages of open-source operating systems. Include the types of people who would find each aspect to be an advantage or a disadvantage.

Advantages:

- **Free and cost-effective** - Ideal for students, hobbyists, and small businesses with limited budgets.

- **Highly customizable** - Developers and tech-savvy users can modify the OS to suit specific needs.
- **Community support and transparency** - Useful for learners and researchers who benefit from openly available source code and forums.
- **Enhanced security (if maintained well)** - Security experts appreciate the ability to audit and patch code quickly.

Disadvantages:

- **Steeper learning curve** - Non-technical users may struggle with setup and troubleshooting.
- **Limited official support** - Businesses or institutions might prefer commercial OSes with professional support.
- **Software compatibility issues** - Gamers or professionals who rely on specific proprietary software may face challenges.
- **Hardware driver issues** - General users might encounter problems with hardware not well-supported by open-source drivers.

Open-source OSes are best for enthusiasts, developers, and budget-conscious users; less ideal for casual or enterprise-level users.

Q-3. Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.

Protecting the operating system by placing it in a fixed, non-modifiable memory partition provided basic security, but it also introduced some significant difficulties:

1. **Lack of Flexibility for Updates or Fixes:**
   If the operating system code resides in a memory partition that cannot be modified, any bugs, security vulnerabilities, or needed feature updates cannot be patched without replacing the entire system. This would require either physically replacing ROM chips or rebooting into a different bootloader-a tedious and impractical process. For developers or system administrators, this severely limits the ability to maintain or improve the system over time.

2. **Inability to Support Dynamic Operating System Behavior:**
   Modern operating systems rely on the ability to load and unload modules dynamically (like device drivers or security patches). A non-modifiable OS

memory partition prevents this, making it difficult to support new hardware or respond to runtime conditions. This rigidity would be especially problematic for system engineers or businesses needing adaptable, scalable systems for evolving technology needs.

Such a scheme ultimately restricts innovation, maintenance, and responsiveness in system design.

Q-4. Keeping in mind the various definitions of operating system, consider whether the operating system should include applications such as web browsers and mail programs. Argue both that it should and that it should not, and support your answers.

Arguments for including applications (like web browsers & mail programs) in the OS:

- Immediate usability: Users can start browsing or emailing right after setup, with no extra installations needed.
- Better system integration: Pre-installed apps can be optimized for the OS, resulting in better speed, performance, and security.
- User convenience: Non-technical users benefit from having essential tools readily available.
- Standardization: Makes it easier for schools, businesses, and governments to manage and support devices with uniform software.

Arguments against including applications in the OS:

- Not part of core OS function: The OS's job is to manage hardware and system resources-not to dictate user software choices.
- Anti-competitive concerns: Bundling apps (e.g., Microsoft with Internet Explorer) can reduce fair competition.
- System bloat: Unwanted apps take up storage and may slow down the system.
- Limits user choice: Advanced users or developers may prefer lighter alternatives or open-source apps.

Conclusion:
Including apps benefits everyday users through convenience, but keeping the OS lean and modular respects user freedom and encourages competition. The decision should depend on the intended user base.

Q-5. Describe the differences between symmetric and asymmetric multiprocessing.

Symmetric multiprocessing (SMP) and **asymmetric multiprocessing (AMP)** are two approaches to using multiple processors in a system, but they differ in how tasks and control are shared.

1. **Symmetric Multiprocessing (SMP):**
   In SMP, all processors are peers and share a common memory and I/O bus. Each processor runs the same operating system and has equal access to system resources. Tasks are dynamically scheduled across all CPUs, which improves performance, fault tolerance, and load balancing. If one processor fails, others can take over its tasks. SMP is commonly found in modern desktops, servers, and smartphones with multi-core processors.

2. **Asymmetric Multiprocessing (AMP):**
   In AMP, processors are assigned specific roles. One processor (the master) controls the system and assigns tasks to the others (the slaves), which may run limited or specialized instructions. Only the master runs the operating system, while slave processors handle predefined tasks like device control or background processing. AMP is easier to implement and useful in embedded systems where tasks are predictable, such as in medical devices, aerospace systems, or early computer architectures.

In summary**,** SMP is flexible and efficient for general-purpose computing, while AMP is suited for systems with dedicated tasks and simpler coordination needs.