1. Internal Linux Commands

a. Display current working directory:

```
pwd
```

b. Navigate to a directory (e.g., Documents):

```
cd ~/Documents
```

c. Verify new location:

```
pwd
```

d. Create a new directory named "TestDir":

```
mkdir TestDir
```

e. Create empty file "testfile.txt" in "TestDir":

```
touch TestDir/testfile.txt
```

f. List contents of "TestDir":

```
ls TestDir
```

g. Delete "testfile.txt" and verify:

```
rm TestDir/testfile.txt
ls TestDir
```

h. Display current system date and time:

```
date
```

i. Show system uptime:

```
uptime
```

j. Display current user:

```
whoami
```

2. Shell Scripts (Any two)

a. Display OS version, release number, kernel version:

```
echo "OS Version and Release Info:"
cat /etc/os-release

echo "Kernel Version:"
uname -r
```

b. Add two numbers:

```
read -p "Enter first number: " a
read -p "Enter second number: " b
sum=$((a + b))
echo "Sum: $sum"
```

c. Even or Odd check:

```
read -p "Enter a number: " num
if (( num % 2 == 0 ))
then
    echo "$num is Even"
else
    echo "$num is Odd"
fi
```

d. Highest memory usage processes:

```
ps aux --sort=-%mem | head -n 5
```

e. Top 10 processes by memory usage (descending):

```
ps aux --sort=-%mem | head -n 11
```

3. Parent fork:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(){
   pid_t pid=fork();

   if(pid==0){
      printf("child Process: PID=%d ,Parent PID=%d\n",getpid(),getppid());
   }
   else if(pid>0){
      printf("Parent Process: PID=%d\n",getpid());
   }
   else{
      printf("fork Failed.\n");
   }

   return 0;
}
```

4. Preemptive Process Scheduling – SJF (C Code Simplified)

```
#include <stdio.h>

int main() {
    int n, i, j;
    printf("Enter number of processes: ");
    scanf("%d", &n);

    int bt[n], p[n], wt[n], tat[n], total_wt = 0, total_tat = 0;

    printf("Enter burst times:\n");
    for (i = 0; i < n; i++) {
        printf("P%d: ", i+1);
        scanf("%d", &bt[i]);
        p[i] = i + 1;
    }
```

```
    // Sort based on burst time
    for (i = 0; i < n-1; i++) {
        for (j = i+1; j < n; j++) {
            if (bt[i] > bt[j]) {
                int temp = bt[i]; bt[i] = bt[j]; bt[j] = temp;
                temp = p[i]; p[i] = p[j]; p[j] = temp;
            }
        }
    }

    wt[0] = 0;
    for (i = 1; i < n; i++) {
        wt[i] = bt[i-1] + wt[i-1];
        total_wt += wt[i];
    }

    for (i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
        total_tat += tat[i];
    }

    printf("P\tBT\tWT\tTAT\n");
    for (i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\n", p[i], bt[i], wt[i], tat[i]);
    }

    return 0;
}
```

5. Disk Scheduling – FCFS (C Code)

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int n, head, i, total = 0;
```

```
    printf("Enter number of requests: ");
    scanf("%d", &n);

    int req[n];
    printf("Enter request sequence: ");
    for (i = 0; i < n; i++)
        scanf("%d", &req[i]);

    printf("Enter initial head position: ");
    scanf("%d", &head);

    for (i = 0; i < n; i++) {
        total += abs(req[i] - head);
        head = req[i];
    }

    printf("Total Head Movement: %d\n", total);
    return 0;
}
```