OPERATING SYSTEMS LAB
DIGITAL ASSIGNMENT-1

AADITYA ROSHAN
22BIT0250

a) Study of Linux commands

1. cat

```
aaditya@AadityaPC:~/OS_Lab$ cat > file1
Lab Assignment-1
Aaditya Roshan
22BIT0250
aaditya@AadityaPC:~/OS_Lab$ cat file1
Lab Assignment-1
Aaditya Roshan
22BIT0250
```

2. cp

```
aaditya@AadityaPC:~/OS_Lab$ cp file1 file2
aaditya@AadityaPC:~/OS_Lab$ cat file2
Lab Assignment-1
Aaditya Roshan
22BIT0250
aaditya@AadityaPC:~/OS_Lab$ cat file1
Lab Assignment-1
Aaditya Roshan
22BIT0250
```

3. head

```
aaditya@AadityaPC:~/OS_Lab$ head file1
Lab Assignment-1
Aaditya Roshan
22BIT0250
```

4. tail

```
aaditya@AadityaPC:~/OS_Lab$ tail file1
Lab Assignment-1
Aaditya Roshan
22BIT0250
```

5. mv

```
aaditya@AadityaPC:~/OS_Lab$ cat file3
Lab Assignment-1
Aaditya Roshan
22BIT0250
aaditya@AadityaPC:~/OS_Lab$ cat file1
cat: file1: No such file or directory
```

6. tty

```
aaditya@AadityaPC:~/OS_Lab$ tty
/dev/pts/0
```

7. mkdir

```
aaditya@AadityaPC:~/OS_Lab$ mkdir dir1
aaditya@AadityaPC:~/OS_Lab$ cd dir1
aaditya@AadityaPC:~/OS_Lab/dir1$ cd
aaditya@AadityaPC:~$ ls
OS_Lab
aaditya@AadityaPC:~$ cd OS_Lab
aaditya@AadityaPC:~/OS_Lab$
```

8. rm

```
aaditya@AadityaPC:~/OS_Lab$ rm file3
aaditya@AadityaPC:~/OS_Lab$ ls
1b.sh  2b.sh  3b.sh  4b.sh  5b.sh  process
```

9. rmdir

```
aaditya@AadityaPC:~/OS_Lab$ rmdir dir1
aaditya@AadityaPC:~/OS_Lab$ ls
1b.sh  2b.sh  3b.sh  4b.sh  5b.sh  file2  file3
aaditya@AadityaPC:~/OS_Lab$
```

10. who

```
aaditya@AadityaPC:~/OS_Lab$ who
aaditya  pts/1        2024-09-03 12:14
```

11. pwd

```
aaditya@AadityaPC:~/OS_Lab$ pwd
/home/aaditya/OS_Lab
```

12. date

```
aaditya@AadityaPC:~/OS_Lab$ date
Tue Sep  3 16:18:01 IST 2024
```

13. ps

```
aaditya@AadityaPC:~/OS_Lab$ ps
    PID TTY          TIME CMD
   1195 pts/0    00:00:02 bash
  50527 pts/0    00:00:00 ps
```

14. cal

```
aaditya@AadityaPC:~/OS_Lab$ cal March 2024
     March 2024
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

15. uptime

```
aaditya@AadityaPC:~/OS_Lab$ uptime
 19:46:07 up  3:32,  1 user,  load average: 0.06, 0.03, 0.00
```

16. whoami

```
aaditya@AadityaPC:~/OS_Lab$ whoami
aaditya
```

17. Arithmetic operations

```
aaditya@AadityaPC:~/OS_Lab$ a=3
aaditya@AadityaPC:~/OS_Lab$ b=2
aaditya@AadityaPC:~/OS_Lab$ expr $a '+' $b
5
```

b) Shell Programming

• Handling the command line arguments

CODE:

```
echo "First argument: $1"
echo "Second argument: $2"
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ vi shell1.sh
aaditya@AadityaPC:~/OS_Lab$ bash shell1.sh 21 18
First argument: 21
Second argument: 18
```

• String reversal, multiplication table

>String reversal

CODE:

```
input_string="$1"
reversed_string=""
len=${#input_string}
for (( i=$len-1; i>=0; i-- )); do
   reversed_string="$reversed_string${input_string:$i:1}"
done
echo "Reversed string: $reversed_string"
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ bash shell2.sh "HelloWorld"
Reversed string: dlroWolleH
```

>Multiplication Table

CODE:

```
echo "Enter a number:"
read num
for i in {1..10}
do
   result=$((num * i))
   echo "$num x $i = $result"
done
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ vi shell2.sh
aaditya@AadityaPC:~/OS_Lab$ bash shell2.sh
Enter a number:
9
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
```

• If-Else, Nested If Else, Switch cases in shell

>if-else:

CODE:

```
echo "Enter a number: "
read num
if [ $num -gt 0 ]; then
   echo "The number is positive."
elif [ $num -lt 0 ]; then
   echo "The number is negative."
else
   echo "The number is zero."
fi
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ vi shell3.sh
aaditya@AadityaPC:~/OS_Lab$ bash shell3.sh
Enter a number:
-10
The number is negative.
aaditya@AadityaPC:~/OS_Lab$ bash shell3.sh
Enter a number:
21
The number is positive.
```

>Nested if-else:

CODE:

```
echo "Enter a number: "
read num
if [ $num -ge 0 ]; then
   if [ $num -le 50 ]; then
      echo "The number is between 0 and 50."
   else
      echo "The number is greater than 50."
   fi
else
   echo "The number is negative."
fi
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ vi shell3.
aaditya@AadityaPC:~/OS_Lab$ bash shell
Enter a number:
25
The number is between 0 and 50.
```

>Switch:

CODE:

```
echo "Do you want to continue? (y/n):"
read answer
case $answer in
   y|Y)
      echo "You chose to continue."
      ;;
   n|N)
      echo "You chose not to continue."
      ;;
   *)
      echo "Invalid choice. Please enter y or n."
      ;;
esac
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ vi shell3.sh
aaditya@AadityaPC:~/OS_Lab$ bash shell3.sh
Do you want to continue? (y/n):
Y
You chose to continue.
aaditya@AadityaPC:~/OS_Lab$ bash shell3.sh
Do you want to continue? (y/n):
n
You chose not to continue.
aaditya@AadityaPC:~/OS_Lab$ bash shell3.sh
Do you want to continue? (y/n):
X
Invalid choice. Please enter y or n.
```

EXERCISES:

1. Read three numbers from the keyboard and print the minimum value.

CODE:

```
echo "Enter the first number"
read num1
echo "Enter the second number"
read num2
echo "Enter the third number"
read num3
min_num=$num1
if [ $num2 -lt $min_num ]; then
      min_num=$num2
fi
if [ $num3 -lt $min_num ]; then
      min_num=$num3
fi
echo "Minimum Number is $min_num"
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ sh 1b.sh
Enter the first number
3
Enter the second number
8
Enter the third number
1
Minimum Number is 1
```

2. Read in three numbers from the keyboard and print the maximum value.

CODE:

```
echo "Enter the first number"
read num1
echo "Enter the second number"
read num2
echo "Enter the third number"
read num3
max_num=$num1
if [ $num2 -gt $max_num ]; then
      max_num=$num2
fi
if [ $num3 -gt $max_num ]; then
      max_num=$num3
fi
echo "Maximum Number is $max_num"
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ sh 2b.sh
Enter the first number
3
Enter the second number
8
Enter the third number
2
Maximum Number is 8
```

3. Swap two numbers without using third variable.

CODE:

```
echo "Enter the first number"
read num1
echo "Enter the second number"
read num2
echo "Numbers before swapping: "
echo "Number 1 : $num1"
echo "Number 2 : $num2"
num1=$num1 + $num2
num2=$num1 - $num2
num1=$num1 - $num2
echo "Numbers after swapping: "
echo "Number 1 : $num1"
echo "Number 2 : $num2"
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ sh 3b.sh
Enter the first number
11
Enter the second number
7
Numbers before swapping:
Number 1 : 11
Number 2 : 7
Numbers after swapping:
Number 1 : 7
Number 2 : 11
```

4. Read the marks and print the grade of the student (use elif).

CODE:

```
echo "Enter the marks of the student:"
read marks
if [ $marks -ge 90 ] && [ $marks -le 100 ]; then
   grade="S"
elif [ $marks -ge 80 ] && [ $marks -lt 90 ]; then
   grade="A"
elif [ $marks -ge 70 ] && [ $marks -lt 80 ]; then
   grade="B"
elif [ $marks -ge 60 ] && [ $marks -lt 70 ]; then
   grade="C"
elif [ $marks -ge 50 ] && [ $marks -lt 60 ]; then
   grade="D"
elif [ $marks -ge 40 ] && [ $marks -lt 50 ]; then
   grade="E"
else
   grade="F"
fi
echo "The grade is $grade"
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ sh 4b.
Enter the marks of the student:
85
The grade is A
aaditya@AadityaPC:~/OS_Lab$ sh 4b.
Enter the marks of the student:
73
The grade is B
```

5. Read two data and perform basic arithmetic operations based on User choice (use case).

CODE:

```
echo "Enter the first number:"
read num1
echo "Enter the second number:"
read num2
echo "Choose an operation:"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read choice
case $choice in
   1)
      result=$((num1 + num2))
      echo "Result of Addition: $result"
      ;;
   2)
      result=$((num1 - num2))
      echo "Result of Subtraction: $result"
      ;;
   3)
      result=$((num1 * num2))
      echo "Result of Multiplication: $result"
      ;;
   4)
      if [ $num2 -ne 0 ]; then
         result=$((num1 / num2))
         echo "Result of Division: $result"
```

```
        else
            echo "Error: Division by zero is not allowed"
        fi
        ;;
    *)
        echo "Invalid choice. Please select a valid operation."
        ;;
esac
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ sh 5b.sh
Enter the first number:
15
Enter the second number:
4
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
3
Result of Multiplication: 60
aaditya@AadityaPC:~/OS_Lab$ sh 5b.sh
Enter the first number:
6
Enter the second number:
0
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
4
Error: Division by zero is not allowed
```

c) Parent child process creation using fork( ) and exec() system call

Checking the Process Identifier
Assigning new task to child
Providing the path name and program name to exec()
Synchronizing Parent and child process using wait()

CODE:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
   pid_t pid;
   pid = fork();
   if (pid < 0) {
      fprintf(stderr, "Fork failed!\n");
      return 1;
   } else if (pid == 0) {
      printf("Child process created with PID: %d\n", getpid());
      execlp("/bin/ls", "ls", NULL);
      fprintf(stderr, "Exec failed!\n");
      return 1;
   } else {
      printf("Parent process PID: %d, Child process PID: %d\n", getpid(), pid);
      wait(NULL);
      printf("Child process has finished execution.\n");
   }

   return 0;
}
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ gcc process.c -o process
aaditya@AadityaPC:~/OS_Lab$ ./process
Parent process PID: 28493, Child process PID: 28494
Child process created with PID: 28494
1b.sh  2b.sh  3b.sh  4b.sh  5b.sh  process  process.c
Child process has finished execution.
```

d) CPU Scheduling
> FCFS Scheduling

```
#include <stdio.h>
typedef struct {
    int id;
    int arrival;
    int burst;
    int waiting;
    int turnaround;
} Process;
void calculateTimes(Process processes[], int n) {
    int total_wt = 0, total_tat = 0;
    int current_time = 0;
    for (int i = 0; i < n; i++) {
        if (current_time < processes[i].arrival) {
            current_time = processes[i].arrival;
        }
        processes[i].waiting = current_time - processes[i].arrival;
        current_time += processes[i].burst;
        processes[i].turnaround = current_time - processes[i].arrival;
        total_wt += processes[i].waiting;
        total_tat += processes[i].turnaround;
    }
    double avg_wt = (double)total_wt / n;
    double avg_tat = (double)total_tat / n;
    printf("Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
               processes[i].id, processes[i].arrival,
               processes[i].burst, processes[i].waiting,
               processes[i].turnaround);
    }
    printf("\nAverage Waiting Time: %.2f\n", avg_wt);
    printf("Average Turnaround Time: %.2f\n", avg_tat);
}
int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    Process processes[n];
    printf("Enter arrival time and burst time for each process:\n");
    for (int i = 0; i < n; i++) {
        processes[i].id = i + 1;
        printf("Process %d - Arrival Time: ", i + 1);
        scanf("%d", &processes[i].arrival);
        printf("Process %d - Burst Time: ", i + 1);
        scanf("%d", &processes[i].burst);
    }

    calculateTimes(processes, n);

    return 0;
}
```

```
aaditya@AadityaPC:~/OS_Lab$ vi fcfs.c
aaditya@AadityaPC:~/OS_Lab$ gcc fcfs.c -o fcfs
aaditya@AadityaPC:~/OS_Lab$ ./fcfs
Enter the number of processes: 4
Enter arrival time and burst time for each process:
Process 1 - Arrival Time: 0
Process 1 - Burst Time: 8
Process 2 - Arrival Time: 2
Process 2 - Burst Time: 6
Process 3 - Arrival Time: 2
Process 3 - Burst Time: 7
Process 4 - Arrival Time: 1
Process 4 - Burst Time: 5
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
1               0               8               0               8
2               2               6               6               12
3               2               7               12              19
4               1               5               20              25

Average Waiting Time: 9.50
Average Turnaround Time: 16.00
```

>SJF Scheduling

```
#include <stdio.h>
#include <limits.h>
typedef struct {
   int id;
   int arrival;
   int burst;
   int waiting;
   int turnaround;
   int remaining;
} Process;
void calculateTimesNonPreemptive(Process processes[], int n) {
   int total_wt = 0, total_tat = 0;
   int completed = 0, current_time = 0;
   int shortest = 0;
   while (completed < n) {
      int min_burst = INT_MAX;
      int idx = -1;
      for (int i = 0; i < n; i++) {
         if (processes[i].arrival <= current_time && processes[i].waiting == -1) {
            if (processes[i].burst < min_burst) {
               min_burst = processes[i].burst;
               idx = i;
            }
         }
      }
      if (idx != -1) {
         processes[idx].waiting = current_time - processes[idx].arrival;
         processes[idx].turnaround = processes[idx].waiting + processes[idx].burst;
         current_time += processes[idx].burst;
         total_wt += processes[idx].waiting;
         total_tat += processes[idx].turnaround;
         completed++;
      } else {
         current_time++;
      }
   }
   // Calculate averages
   double avg_wt = (double)total_wt / n;
   double avg_tat = (double)total_tat / n;
   printf("Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");
   for (int i = 0; i < n; i++) {
      printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
            processes[i].id, processes[i].arrival,
            processes[i].burst, processes[i].waiting,
            processes[i].turnaround);
   }
   printf("\nAverage Waiting Time: %.2f\n", avg_wt);
   printf("Average Turnaround Time: %.2f\n", avg_tat);
}
void calculateTimesPreemptive(Process processes[], int n) {
   int total_wt = 0, total_tat = 0;
   int current_time = 0, completed = 0;
   int min_remaining = INT_MAX;
   int idx = -1;
   while (completed < n) {
      min_remaining = INT_MAX;
      idx = -1;
      for (int i = 0; i < n; i++) {
         if (processes[i].arrival <= current_time && processes[i].remaining > 0) {
```

```
            if (processes[i].remaining < min_remaining) {
                min_remaining = processes[i].remaining;
                idx = i;
            }
        }
    }
    if (idx != -1) {
        processes[idx].remaining--;
        if (processes[idx].remaining == 0) {
            processes[idx].turnaround = current_time + 1 - processes[idx].arrival;
            processes[idx].waiting = processes[idx].turnaround - processes[idx].burst;
            total_wt += processes[idx].waiting;
            total_tat += processes[idx].turnaround;
            completed++;
        }
    }
    current_time++;
}
double avg_wt = (double)total_wt / n;
double avg_tat = (double)total_tat / n;
printf("Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");
for (int i = 0; i < n; i++) {
    printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
           processes[i].id, processes[i].arrival,
           processes[i].burst, processes[i].waiting,
           processes[i].turnaround);
}

printf("\nAverage Waiting Time: %.2f\n", avg_wt);
printf("Average Turnaround Time: %.2f\n", avg_tat);
}
int main() {
int n, choice;
printf("Enter the number of processes: ");
scanf("%d", &n);
Process processes[n];
printf("Enter arrival time and burst time for each process:\n");
for (int i = 0; i < n; i++) {
    processes[i].id = i + 1;
    processes[i].waiting = -1;
    printf("Process %d - Arrival Time: ", i + 1);
    scanf("%d", &processes[i].arrival);
    printf("Process %d - Burst Time: ", i + 1);
    scanf("%d", &processes[i].burst);
    processes[i].remaining = processes[i].burst;
}
printf("\nSelect Scheduling Type:\n");
printf("1. Non-Preemptive SJF\n");
printf("2. Preemptive SJF\n");
printf("Enter your choice (1 or 2): ");
scanf("%d", &choice);
switch (choice) {
    case 1:
        calculateTimesNonPreemptive(processes, n);
        break;
    case 2:
        calculateTimesPreemptive(processes, n);
        break;
    default:
        printf("Invalid choice.\n");
```

```
        break;
    }
    return 0;
}
```

-Non-preemptive (SJF)

```
aaditya@AadityaPC:~/OS_Lab$ vi sjf.c
aaditya@AadityaPC:~/OS_Lab$ gcc sjf.c -o sjf
aaditya@AadityaPC:~/OS_Lab$ ./sjf
Enter the number of processes: 4
Enter arrival time and burst time for each process:
Process 1 - Arrival Time: 0
Process 1 - Burst Time: 7
Process 2 - Arrival Time: 3
Process 2 - Burst Time: 6
Process 3 - Arrival Time: 2
Process 3 - Burst Time: 6
Process 4 - Arrival Time: 2
Process 4 - Burst Time: 9

Select Scheduling Type:
1. Non-Preemptive SJF
2. Preemptive SJF
Enter your choice (1 or 2): 1
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
1               0               7               0               7
2               3               6               4               10
3               2               6               11              17
4               2               9               17              26

Average Waiting Time: 8.00
Average Turnaround Time: 15.00
```

-Preemptive (SJF)

```
aaditya@AadityaPC:~/OS_Lab$ ./sjf
Enter the number of processes: 4
Enter arrival time and burst time for each process:
Process 1 - Arrival Time: 2
Process 1 - Burst Time: 5
Process 2 - Arrival Time: 1
Process 2 - Burst Time: 7
Process 3 - Arrival Time: 4
Process 3 - Burst Time: 8
Process 4 - Arrival Time: 2
Process 4 - Burst Time: 6

Select Scheduling Type:
1. Non-Preemptive SJF
2. Preemptive SJF
Enter your choice (1 or 2): 2
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround Time
1               2               5               0               5
2               1               7               5               12
3               4               8               15              23
4               2               6               11              17

Average Waiting Time: 7.75
Average Turnaround Time: 14.25
```

>Priority Scheduling

```
#include <stdio.h>
#include <limits.h>
void findWaitingTimeNonPreemptive(int processes[], int n, int bt[], int priority[], int arrival[], int wt[]) {
   int completion_time[n], turnaround_time[n];
   int total_wt = 0, total_tat = 0;
   int current_time = 0;
   // Initialize waiting times and completion times
   for (int i = 0; i < n; i++) {
      completion_time[i] = 0;
      wt[i] = 0;
   }
   // Sorting processes by priority (lower number means higher priority)
   for (int i = 0; i < n - 1; i++) {
      for (int j = i + 1; j < n; j++) {
         if (priority[i] > priority[j]) {
            int temp = bt[i];
            bt[i] = bt[j];
            bt[j] = temp;
            temp = priority[i];
            priority[i] = priority[j];
            priority[j] = temp;
            temp = arrival[i];
            arrival[i] = arrival[j];
            arrival[j] = temp;
            temp = processes[i];
            processes[i] = processes[j];
            processes[j] = temp;
         }
      }
   }
   // Calculate waiting time and turnaround time
   for (int i = 0; i < n; i++) {
      if (i == 0) {
         current_time = arrival[i];
         wt[i] = 0;
         current_time += bt[i];
      } else {
         if (current_time < arrival[i]) {
            current_time = arrival[i];
         }
         wt[i] = current_time - arrival[i];
         current_time += bt[i];
      }
      turnaround_time[i] = wt[i] + bt[i];
   }
   printf("Process\tBurst Time\tPriority\tArrival Time\tWaiting Time\tTurnaround Time\n");
   for (int i = 0; i < n; i++) {
      printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i], bt[i], priority[i], arrival[i], wt[i], turnaround_time[i]);
   }
   for (int i = 0; i < n; i++) {
      total_wt += wt[i];
      total_tat += turnaround_time[i];
   }
   printf("\nAverage waiting time = %.2f", (float)total_wt / n);
   printf("\nAverage turnaround time = %.2f\n", (float)total_tat / n);
}
void findWaitingTimePreemptive(int processes[], int n, int bt[], int priority[], int arrival[], int wt[]) {
   int rem_bt[n], time = 0;
   int min_priority, min_index;
```

```
    int completed = 0;
    // Initialize remaining burst times
    for (int i = 0; i < n; i++)
        rem_bt[i] = bt[i];
    // Initialize waiting times
    for (int i = 0; i < n; i++)
        wt[i] = 0;
    while (completed < n) {
        min_priority = INT_MAX;
        min_index = -1;
        // Find the process with the highest priority that has arrived
        for (int i = 0; i < n; i++) {
            if (rem_bt[i] > 0 && arrival[i] <= time && priority[i] < min_priority) {
                min_priority = priority[i];
                min_index = i;
            }
        }
        // If no process is available, advance time
        if (min_index == -1) {
            time++;
            continue;
        }
        rem_bt[min_index]--;
        time++;
        // Update waiting time for other processes
        for (int i = 0; i < n; i++) {
            if (i != min_index && rem_bt[i] > 0 && arrival[i] <= time) {
                wt[i]++;
            }
        }
        if (rem_bt[min_index] == 0) {
            completed++;
        }
    }
    printf("Process\tBurst Time\tPriority\tArrival Time\tWaiting Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", processes[i], bt[i], priority[i], arrival[i], wt[i]);
    }
    int total_wt = 0;
    for (int i = 0; i < n; i++)
        total_wt += wt[i];
    printf("\nAverage waiting time = %.2f\n", (float)total_wt / n);
}
int main() {
    int n;
    int choice;
    // Input number of processes
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    int processes[n], bt[n], priority[n], arrival[n], wt[n];
    // Input burst times, priorities, and arrival times
    printf("Enter burst times for %d processes:\n", n);
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1; // Assigning process IDs starting from 1
        printf("Process %d: ", processes[i]);
        scanf("%d", &bt[i]);
    }
    printf("Enter priorities for %d processes:\n", n);
    for (int i = 0; i < n; i++) {
        printf("Priority of Process %d: ", processes[i]);
```

```
        scanf("%d", &priority[i]);
    }
    printf("Enter arrival times for %d processes:\n", n);
    for (int i = 0; i < n; i++) {
        printf("Arrival time of Process %d: ", processes[i]);
        scanf("%d", &arrival[i]);
    }
    printf("Select scheduling type:\n1. Non-Preemptive\n2. Preemptive\n");
    scanf("%d", &choice);
    if (choice == 1) {
        findWaitingTimeNonPreemptive(processes, n, bt, priority, arrival, wt);
    } else if (choice == 2) {
        findWaitingTimePreemptive(processes, n, bt, priority, arrival, wt);
    } else {
        printf("Invalid choice\n");
    }
    return 0;
}
```

-Non-preemptive (Priority)

```
aaditya@AadityaPC:~/OS_Lab$ vi priority.c
aaditya@AadityaPC:~/OS_Lab$ gcc priority.c -o priority
aaditya@AadityaPC:~/OS_Lab$ ./priority
Enter the number of processes: 4
Enter burst times for 4 processes:
Process 1: 2
Process 2: 6
Process 3: 7
Process 4: 8
Enter priorities for 4 processes:
Priority of Process 1: 2
Priority of Process 2: 1
Priority of Process 3: 4
Priority of Process 4: 3
Enter arrival times for 4 processes:
Arrival time of Process 1: 2
Arrival time of Process 2: 3
Arrival time of Process 3: 0
Arrival time of Process 4: 2
Select scheduling type:
1. Non-Preemptive
2. Preemptive
1
Process Burst Time      Priority        Arrival Time    Waiting Time    Turnaround Time
2       6               1               3               0               6
1       2               2               2               7               9
4       8               3               2               9               17
3       7               4               0               19              26

Average waiting time = 8.75
Average turnaround time = 14.50
```

-Preemptive (Priority)

```
aaditya@AadityaPC:~/OS_Lab$ ./priority
Enter the number of processes: 4
Enter burst times for 4 processes:
Process 1: 7
Process 2: 6
Process 3: 8
Process 4: 4
Enter priorities for 4 processes:
Priority of Process 1: 2
Priority of Process 2: 1
Priority of Process 3: 3
Priority of Process 4: 4
Enter arrival times for 4 processes:
Arrival time of Process 1: 0
Arrival time of Process 2: 2
Arrival time of Process 3: 2
Arrival time of Process 4: 4
Select scheduling type:
1. Non-Preemptive
2. Preemptive
2
Process Burst Time      Priority        Arrival Time    Waiting Time
1       7               2               0               6
2       6               1               2               1
3       8               3               2               12
4       4               4               4               18

Average waiting time = 9.25
```

>Round Robin

```
#include <stdio.h>
void findWaitingTime(int processes[], int n, int bt[], int arrival[], int wt[], int quantum) {
   int rem_bt[n];
   for (int i = 0; i < n; i++)
      rem_bt[i] = bt[i];
   int t = 0; // Current time
   while (1) {
      int done = 1;
      for (int i = 0; i < n; i++) {
         if (rem_bt[i] > 0 && arrival[i] <= t) {
            done = 0;
            if (rem_bt[i] > quantum) {
               t += quantum;
               rem_bt[i] -= quantum;
            } else {
               t = t + rem_bt[i];
               wt[i] = t - bt[i] - arrival[i];
               rem_bt[i] = 0;
            }
         }
      }
      if (done == 1)
         break;
   }
}
void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
   for (int i = 0; i < n; i++)
      tat[i] = bt[i] + wt[i];
}
void findavgTime(int processes[], int n, int bt[], int arrival[], int quantum) {
   int wt[n], tat[n];
   findWaitingTime(processes, n, bt, arrival, wt, quantum);
   findTurnAroundTime(processes, n, bt, wt, tat);
   float total_wt = 0, total_tat = 0;
   for (int i = 0; i < n; i++) {
      total_wt += wt[i];
      total_tat += tat[i];
   }
   printf("Processes  Burst Time  Arrival Time  Waiting Time  Turn-Around Time\n");
   for (int i = 0; i < n; i++)
      printf("  %d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i], bt[i], arrival[i], wt[i], tat[i]);
   printf("\nAverage waiting time = %.5f", total_wt / n);
   printf("\nAverage turn-around time = %.5f", total_tat / n);
}
int main() {
   int n;
   printf("Enter the number of processes: ");
   scanf("%d", &n);
   int processes[n];
   int bt[n], arrival[n];
   int quantum;
   printf("Enter burst times for %d processes:\n", n);
   for (int i = 0; i < n; i++) {
      processes[i] = i + 1;
      printf("Process %d: ", processes[i]);
      scanf("%d", &bt[i]);
   }
   printf("Enter arrival times for %d processes:\n", n);
   for (int i = 0; i < n; i++) {
```

```
    printf("Arrival time of Process %d: ", processes[i]);
    scanf("%d", &arrival[i]);
  }
  printf("Enter time quantum: ");
  scanf("%d", &quantum);
  findavgTime(processes, n, bt, arrival, quantum);
  printf("\n");
  return 0;
}
```

```
aaditya@AadityaPC:~/OS_Lab$ gcc round_robin.c
aaditya@AadityaPC:~/OS_Lab$ gcc round_robin.c -o round_robin
aaditya@AadityaPC:~/OS_Lab$ ./round_robin
Enter the number of processes: 4
Enter burst times for 4 processes:
Process 1: 8
Process 2: 6
Process 3: 10
Process 4: 5
Enter arrival times for 4 processes:
Arrival time of Process 1: 4
Arrival time of Process 2: 1
Arrival time of Process 3: 0
Arrival time of Process 4: 2
Enter time quantum: 3
Processes  Burst Time  Arrival Time  Waiting Time  Turn-Around Time
  1             8              4              16              24
  2             6              1              16              22
  3             10             0              19              29
  4             5              2              10              15

Average waiting time = 15.25000
Average turn-around time = 22.50000
```

iii. Consider a corporate hospital where we have n number of patients waiting for consultation. The amount of time required to serve a patient may vary, say 10 to 30 minutes. If a patient arrives with an emergency, he /she should be attended immediately before other patients, which may increase the waiting time of other patients. If you are given this problem with the following algorithms, how would you devise an effective scheduling so that it optimizes the overall performance such as minimizing the waiting time of all patients. [Single queue or multi-level queue can be used].

CODE:

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

#define MAX_PATIENTS 100

typedef struct {
    int id;
    char name[50];
    int age;
    char gender;
    int arrival_time;
    int consultation_time;
    int priority;
    int review;
} Patient;

Patient priorityQueue[MAX_PATIENTS];
int pq_size = 0;

void swap(Patient *a, Patient *b) {
    Patient temp = *a;
    *a = *b;
    *b = temp;
}

void heapify(int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;

    if (left < pq_size && priorityQueue[left].priority < priorityQueue[largest].priority)
        largest = left;

    if (right < pq_size && priorityQueue[right].priority < priorityQueue[largest].priority)
        largest = right;

    if (largest != i) {
        swap(&priorityQueue[i], &priorityQueue[largest]);
        heapify(largest);
     }
}
```

```
void insertPatient(Patient p) {
    int urgency = p.priority;
    int consultation_time_factor = p.review ? 1 : 2;
    p.priority = urgency * consultation_time_factor;

    priorityQueue[pq_size++] = p;
    for (int i = pq_size / 2 - 1; i >= 0; i--)
        heapify(i);
}

Patient extractMin() {
    if (pq_size == 0) {
        printf("No patients left to attend.\n");
        exit(1);
    }

    Patient root = priorityQueue[0];
    priorityQueue[0] = priorityQueue[--pq_size];
    heapify(0);
    return root;
}

void schedulePatients(Patient patients[], int n) {
    int current_time = 0;
    int i;

    for (i = 0; i < n; i++) {
        insertPatient(patients[i]);
    }

    while (pq_size > 0) {
        Patient p = extractMin();

        printf("\nAttending Patient ID %d: %s\n", p.id, p.name);
        printf("Priority: %d, Review: %d, Consultation Time: %d mins\n", p.priority, p.review,
p.consultation_time);
printf("Priority: %d, Review: %d, Consultation Time: %d mins\n", p.priority, p.review,
p.consultation_time);
        printf("Start Time: %d mins, End Time: %d mins\n", current_time, current_time +
p.consultation_time);

        current_time += p.consultation_time;
    }
}

int main() {
    int n, i;
    Patient patients[MAX_PATIENTS];
```

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

    for (i = 0; i < n; i++) {
        printf("Enter details for patient %d\n", i + 1);
        printf("ID: ");
        scanf("%d", &patients[i].id);
        printf("Name: ");
        scanf("%s", patients[i].name);
        printf("Age: ");
        scanf("%d", &patients[i].age);
        printf("Gender (M/F): ");
        scanf(" %c", &patients[i].gender);
        printf("Arrival Time (in minutes): ");
        scanf("%d", &patients[i].arrival_time);
        printf("Consultation Time (in minutes): ");
        scanf("%d", &patients[i].consultation_time);
        printf("Priority (1 for Emergency, 2 for High Priority, 3 for Regular): ");
        scanf("%d", &patients[i].priority);
        printf("Review? (1 for Yes, 0 for No): ");
        scanf("%d", &patients[i].review);

        if (patients[i].review) {
            patients[i].consultation_time = rand() % 5 + 5;
        }
    }
schedulePatients(patients, n);

    return 0;
}
```

OUTPUT:

```
aaditya@AadityaPC:~/OS_Lab$ gcc hosp_scheduling.c -o hosp_scheduling
aaditya@AadityaPC:~/OS_Lab$ ./hosp_scheduling
Enter the number of patients: 4
Enter details for patient 1
ID: 111
Name: Abhay
Age: 22
Gender (M/F): M
Arrival Time (in minutes): 15
Consultation Time (in minutes): 12
Priority (1 for Emergency, 2 for High Priority, 3 for Regular): 2
Review? (1 for Yes, 0 for No): 1
Enter details for patient 2
ID: 222
Name: Eshita
Age: 20
Gender (M/F): F
Arrival Time (in minutes): 20
Consultation Time (in minutes): 25
Priority (1 for Emergency, 2 for High Priority, 3 for Regular): 3
Review? (1 for Yes, 0 for No): 0
Enter details for patient 3
ID: 333
Name: Brindha
Age: 25
Gender (M/F): F
Arrival Time (in minutes): 20
Consultation Time (in minutes): 16
Priority (1 for Emergency, 2 for High Priority, 3 for Regular): 1
Review? (1 for Yes, 0 for No): 1
Enter details for patient 4
ID: 444
Name: Danush
Age: 27
Gender (M/F): M
Arrival Time (in minutes): 32
Consultation Time (in minutes): 18
Priority (1 for Emergency, 2 for High Priority, 3 for Regular): 1
Review? (1 for Yes, 0 for No): 0

Attending Patient ID 333: Brindha
Priority: 1, Review: 1, Consultation Time: 6 mins
Start Time: 0 mins, End Time: 6 mins

Attending Patient ID 444: Danush
Priority: 2, Review: 0, Consultation Time: 18 mins
Start Time: 6 mins, End Time: 24 mins

Attending Patient ID 111: Abhay
Priority: 2, Review: 1, Consultation Time: 8 mins
Start Time: 24 mins, End Time: 32 mins

Attending Patient ID 222: Eshita
Priority: 6, Review: 0, Consultation Time: 25 mins
Start Time: 32 mins, End Time: 57 mins
aaditya@AadityaPC:~/OS_Lab$
```