Operating Systems LAB

PAPER CODE	**:**	**CIC-353**
Faculty Name	:	Ms. Kavita Saxena
Name	:	Amit Singhal
Enrollment No.	:	11614802722
Branch	:	Computer Science & Engg.
Semester | Group	:	5C6

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

MAIT

उद्यमेन हि सिध्यन्ति
कार्याणि न मनोरथैः

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY
PSP Area, Plot No. 1, Sector-22, Rohini, Delhi-110086

MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

Computer Science & Engineering Department

VISION

"To be centre of excellence in education, research and technology transfer in the field of computer engineering and promote entrepreneurship and ethical values."

MISSION

To foster an open, multidisciplinary and highly collaborative research environment for producing world-class engineers capable of providing innovative solutions to real-life problems and fulfil societal needs.

LAB Assessment Sheet

S.No.	Experiment Name	M	A	R	K	S	Total Marks	Date of Perf.	Date of Check.	Signature
		R1	R2	R3	R4	R5				

S.No.	Experiment Name	M	A	R	K	S	Total Marks	Date of Perf.	Date of Check.	Signature
		R1	R2	R3	R4	R5				

Lab Exercise - 1

❖ AIM :: Introduction to Linux & vi-Editor

1. Introduction to Linux

- **What is Linux?**: Linux is a powerful and versatile open-source operating system based on the Unix architecture. It was created by Linus Torvalds in 1991 and has since grown into a widely-used platform for both personal and professional computing.
- **Open Source Nature**: One of the defining characteristics of Linux is that its source code is freely available for anyone to view, modify, and distribute. This has led to a collaborative environment where developers worldwide contribute to its development.
- **Kernel and Distributions**: Linux is composed of a kernel, which is the core component of the OS, and various distributions (distros) that bundle the kernel with software and package management systems. Popular distributions include Ubuntu, Fedora, Debian, and CentOS.
- **Linux in Different Environments**: Linux is used in a variety of environments, including desktops, servers, mobile devices, and embedded systems. Its flexibility allows it to run on a wide range of hardware, from supercomputers to small IoT devices.

2. Overview of the `vi` Editor

The `vi` (Visual Editor) is a powerful text editor available on almost all Unix-like operating systems, including Linux. It's known for its efficiency and versatility, particularly in environments where only a terminal interface is available. Here is a detailed look at the `vi` editor and its commands, presented in informative points.

---

1. Basics of `vi` Editor

- **Launching** `vi`: To start `vi`, type `vi filename` in the terminal. If `filename` does not exist, `vi` will create it.
- **Modes in** `vi`:
  - **Normal Mode**: The default mode where you can navigate and manipulate text.
  - **Insert Mode**: Used for inserting text. Enter by pressing `i`, `a`, or `o`.
  - **Command Mode**: Enter by typing `:` in Normal Mode for commands like saving, quitting, etc.
  - **Visual Mode**: Used to highlight and manipulate blocks of text.

2. Basic Commands for Running a C File

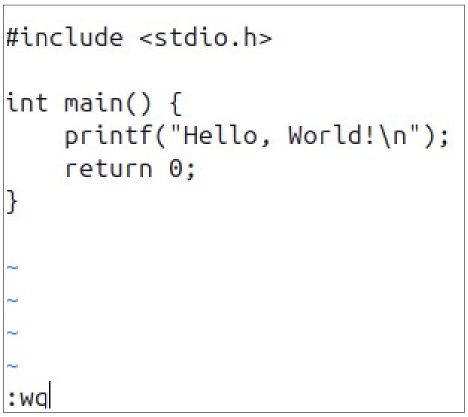
To work with C files in the `vi` editor, you only need a few basic commands to edit, save, and compile the file. Here’s a simplified guide:

- **Open a File**: `vi filename.c`
    - Launches `vi` and opens the file named `filename.c`. If it doesn't exist, `vi` will create it.
- **Insert Mode**:
    - `i`: Enter Insert Mode before the cursor position.
    - `I`: Enter Insert Mode at the beginning of the line.
    - a: Enter Insert Mode after the cursor position.
    - A: Enter Insert Mode at the end of the line.
    - o: Open a new line below the current line and enter Insert Mode.
    - O: Open a new line above the current line and enter Insert Mode.
- **Save and Exit**:
    - `:w`: Save the file without exiting.
    - `:w filename`: Save the file with a new name.
    - `:q`: Quit `vi` without saving.
    - `:wq` **or** ZZ: Save the file and quit `vi`.
    - `:q!`: Quit without saving changes.

Implementation

Writing and Running a basic "Hello, World!" program in C using the terminal on a Linux system.

```
1. cd ~/project

2. vi hello.c
```



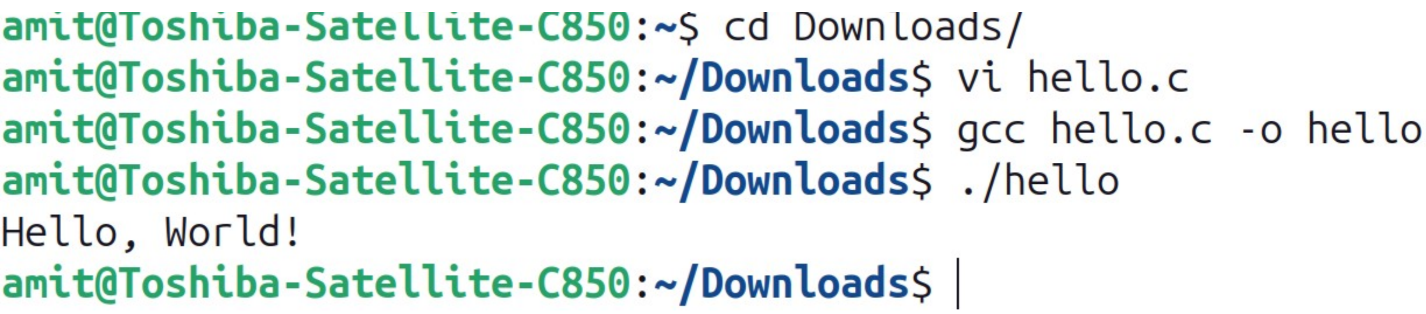
```
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}

~
~
~
~
:wq
```

`/*` **`Save and Exit`** `vi:`

- Press `Esc` to exit Insert Mode.
- Type `:wq` and press `Enter` to save the file and quit `vi`.

`*/`

```
3. gcc hello.c -o hello

4. ./hello
```



```
amit@Toshiba-Satellite-C850:~$ cd Downloads/
amit@Toshiba-Satellite-C850:~/Downloads$ vi hello.c
amit@Toshiba-Satellite-C850:~/Downloads$ gcc hello.c -o hello
amit@Toshiba-Satellite-C850:~/Downloads$ ./hello
Hello, World!
amit@Toshiba-Satellite-C850:~/Downloads$
```

Lab Exercise - 2

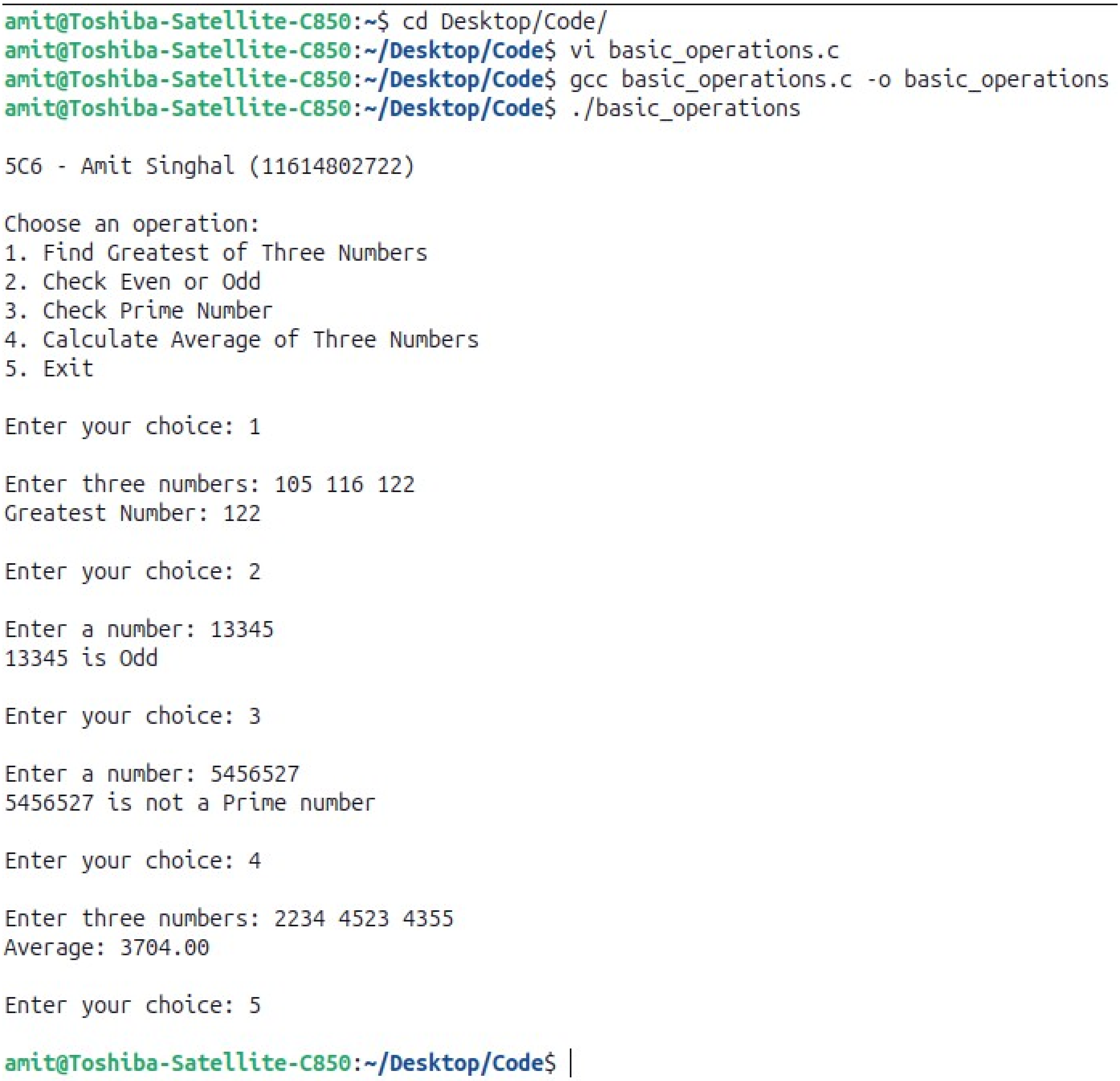
- AIM :: WAP in C to implement basic operations in different functions on Linux using vi-Editor

Source_Code ::

```
#include <stdio.h>

// Function to find the greatest number among three numbers

int findGreatest(int a, int b, int c)

{

    if (a > b && a > c) {

        return a;

    } else if (b > c) {

        return b;

    } else {

        return c;

    }

}

// Function to check if a number is even or odd

void evenOdd(int num)

{

    if (num % 2 == 0) {

        printf("%d is Even\n", num);

    } else {

        printf("%d is Odd\n", num);

    }

}
```

```
// Function to check if a number is prime
void checkPrime(int num)
{
    int i, flag = 0;
    if (num <= 1) {
        printf("%d is not a Prime number\n", num);
        return;
    }
    for (i = 2; i <= num / 2; ++i) {
        if (num % i == 0) {
            flag = 1;
            break;
        }
    }
    if (flag == 0) {
        printf("%d is a Prime number\n", num);
    } else {
        printf("%d is not a Prime number\n", num);
    }
}
// Function to calculate the average of three numbers
double calculateAverage(int a, int b, int c) { return (a + b + c) / 3.0; }
int main()
{
    printf("\n5C6 - Amit Singhal (11614802722)\n");
    int num1, num2, num3;
    int choice;
    printf("\nChoose an operation:\n");
    printf("1. Find Greatest of Three Numbers\n");
    printf("2. Check Even or Odd\n");
printf("3. Check Prime Number\n");

printf("4. Calculate Average of Three Numbers\n");

printf("5. Exit\n");

while (1) {

    printf("\nEnter your choice: ");

    scanf("%d", &choice);

    switch (choice) {

    case 1:

        printf("\nEnter three numbers: ");

        scanf("%d %d %d", &num1, &num2, &num3);

        printf("Greatest Number: %d\n", findGreatest(num1, num2, num3));

        break;

    case 2:

        printf("\nEnter a number: ");

        scanf("%d", &num1);

        evenOdd(num1);

        break;

    case 3:

        printf("\nEnter a number: ");

        scanf("%d", &num1);

        checkPrime(num1);

        break;

    case 4:

        printf("\nEnter three numbers: ");

        scanf("%d %d %d", &num1, &num2, &num3);

        printf("Average: %.2f\n", calculateAverage(num1, num2, num3));

        break;

    case 5:

        printf("\n");

        return 0;

    default:
            printf("\nInvalid choice! Please choose again.\n");
        }
    }
    return 0;
}
```

Output ::



```
amit@Toshiba-Satellite-C850:~$ cd Desktop/Code/
amit@Toshiba-Satellite-C850:~/Desktop/Code$ vi basic_operations.c
amit@Toshiba-Satellite-C850:~/Desktop/Code$ gcc basic_operations.c -o basic_operations
amit@Toshiba-Satellite-C850:~/Desktop/Code$ ./basic_operations

5C6 - Amit Singhal (11614802722)

Choose an operation:
1. Find Greatest of Three Numbers
2. Check Even or Odd
3. Check Prime Number
4. Calculate Average of Three Numbers
5. Exit

Enter your choice: 1

Enter three numbers: 105 116 122
Greatest Number: 122

Enter your choice: 2

Enter a number: 13345
13345 is Odd

Enter your choice: 3

Enter a number: 5456527
5456527 is not a Prime number

Enter your choice: 4

Enter three numbers: 2234 4523 4355
Average: 3704.00

Enter your choice: 5

amit@Toshiba-Satellite-C850:~/Desktop/Code$ 
```

Lab Exercise – 2.2

❖ AIM :: WAP in C to implement basic operations in different functions on Linux using vi-Editor.

Source_Code ::

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

// Function to print the Fibonacci series up to n terms
void fibonacci(int n)
{
   int first = 0, second = 1, next;

   if (n <= 0) {
      printf("Please enter a positive integer.\n");
      return;
   }

   printf("Fibonacci Series: ");
   for (int i = 1; i <= n; i++) {
      if (i == 1) {
         printf("%d ", first);
         continue;
      }
      if (i == 2) {
         printf("%d ", second);
```

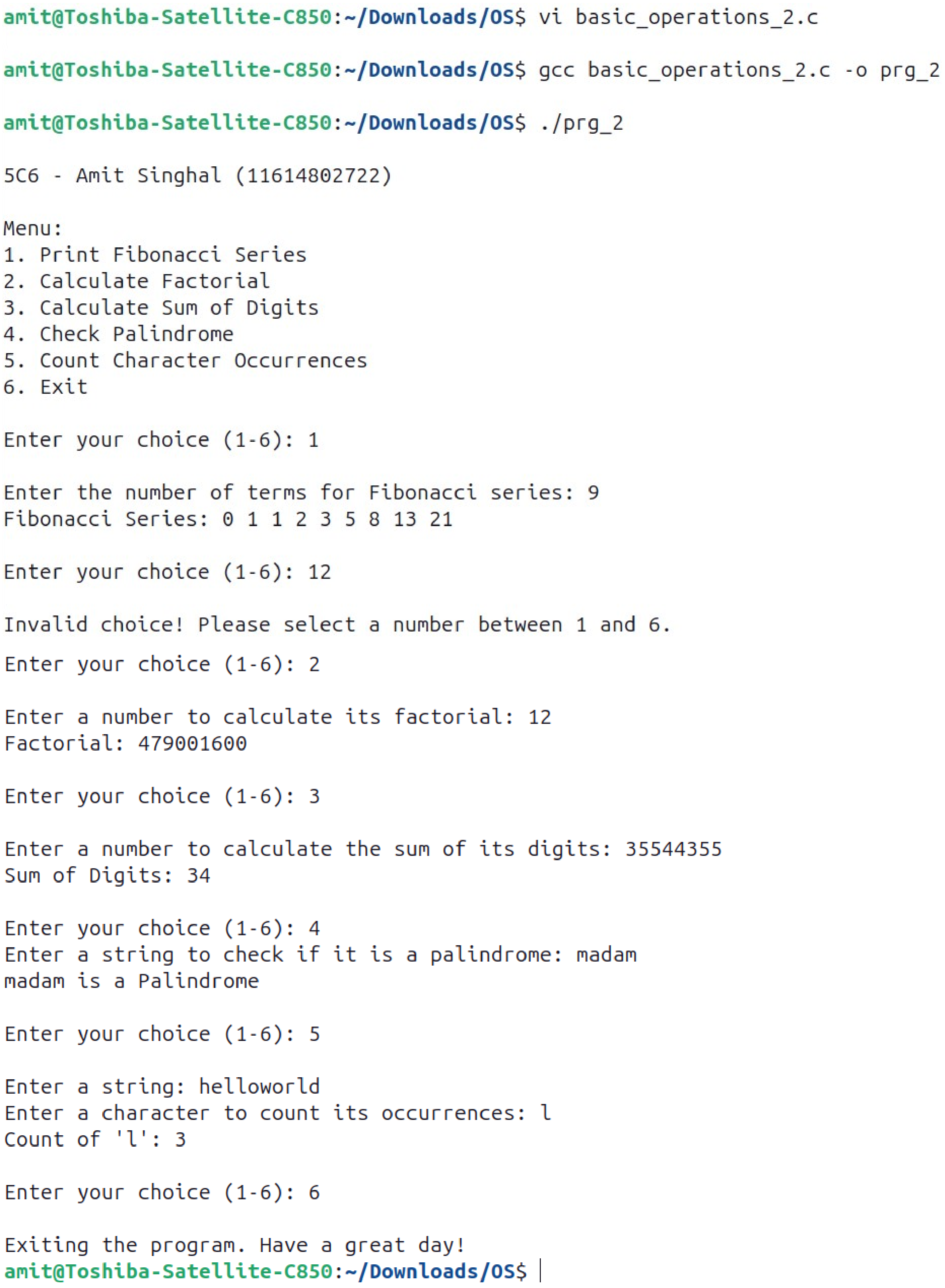
```
            continue;
        }
        next = first + second;
        first = second;
        second = next;
        printf("%d ", next);
    }
    printf("\n");
}

// Function to calculate the factorial of a number
int factorial(int n)
{
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}

// Function to calculate the sum of digits of a number
int digitsSum(int num)
{
    int sum = 0;
    while (num != 0) {
        sum += num % 10;
        num /= 10;
    }
    return sum;
}

// Function to check if a string is a palindrome
bool isPalindrome(char str[])
```

```
{
    int length = strlen(str);
    int start = 0;
    int end = length - 1;

    while (start < end) {
        if (str[start] != str[end]) {
            return false;
        }
        start++;
        end--;
    }
    return true;
}

// Function to count the occurrences of a character in a string
int countChar(char* str, char ch)
{
    int count = 0;
    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] == ch) {
            count++;
        }
    }
    return count;
}

int main()
{

    int choice, num1, num2, num3;
    char str[100], ch;
```

```
printf("\n5C6 - Amit Singhal (11614802722)\n");

// Display the menu
printf("\nMenu:\n");
printf("1. Print Fibonacci Series\n");
printf("2. Calculate Factorial\n");
printf("3. Calculate Sum of Digits\n");
printf("4. Check Palindrome\n");
printf("5. Count Character Occurrences\n");
printf("6. Exit\n");

while (1) {
    printf("\nEnter your choice (1-6): ");
    scanf("%d", &choice);

    switch (choice) {

    case 1:
        printf("\nEnter the number of terms for Fibonacci series: ");
        scanf("%d", &num1);
        fibonacci(num1);
        break;
    case 2:
        printf("\nEnter a number to calculate its factorial: ");
        scanf("%d", &num1);
        printf("Factorial: %d\n", factorial(num1));
        break;
    case 3:
        printf("\nEnter a number to calculate the sum of its digits: ");
        scanf("%d", &num1);
        printf("Sum of Digits: %d\n", digitsSum(num1));
```

```
            break;
        case 4:
            printf("Enter a string to check if it is a palindrome: ");
            scanf("%s", str);

            if (isPalindrome(str)) {
                printf("%s is a Palindrome\n", str);
            } else {
                printf("%s is not a Palindrome\n", str);
            }
            break;
        case 5:
            printf("\nEnter a string: ");
            scanf("%s", str);
            printf("Enter a character to count its occurrences: ");
            scanf(" %c", &ch);
            printf("Count of '%c': %d\n", ch, countChar(str, ch));
            break;
        case 6:
            printf("\nExiting the program. Have a great day!\n");
            return 0;
        default:
            printf(
                "\nInvalid choice! Please select a number between 1 and 6.\n");
        }
    }

    return 0;
}
```

Output ::



```
amit@Toshiba-Satellite-C850:~/Downloads/OS$ vi basic_operations_2.c

amit@Toshiba-Satellite-C850:~/Downloads/OS$ gcc basic_operations_2.c -o prg_2

amit@Toshiba-Satellite-C850:~/Downloads/OS$ ./prg_2

5C6 - Amit Singhal (11614802722)

Menu:
1. Print Fibonacci Series
2. Calculate Factorial
3. Calculate Sum of Digits
4. Check Palindrome
5. Count Character Occurrences
6. Exit

Enter your choice (1-6): 1

Enter the number of terms for Fibonacci series: 9
Fibonacci Series: 0 1 1 2 3 5 8 13 21

Enter your choice (1-6): 12

Invalid choice! Please select a number between 1 and 6.
Enter your choice (1-6): 2

Enter a number to calculate its factorial: 12
Factorial: 479001600

Enter your choice (1-6): 3

Enter a number to calculate the sum of its digits: 35544355
Sum of Digits: 34

Enter your choice (1-6): 4
Enter a string to check if it is a palindrome: madam
madam is a Palindrome

Enter your choice (1-6): 5

Enter a string: helloworld
Enter a character to count its occurrences: l
Count of 'l': 3

Enter your choice (1-6): 6

Exiting the program. Have a great day!
amit@Toshiba-Satellite-C850:~/Downloads/OS$
```

Lab Exercise – 3

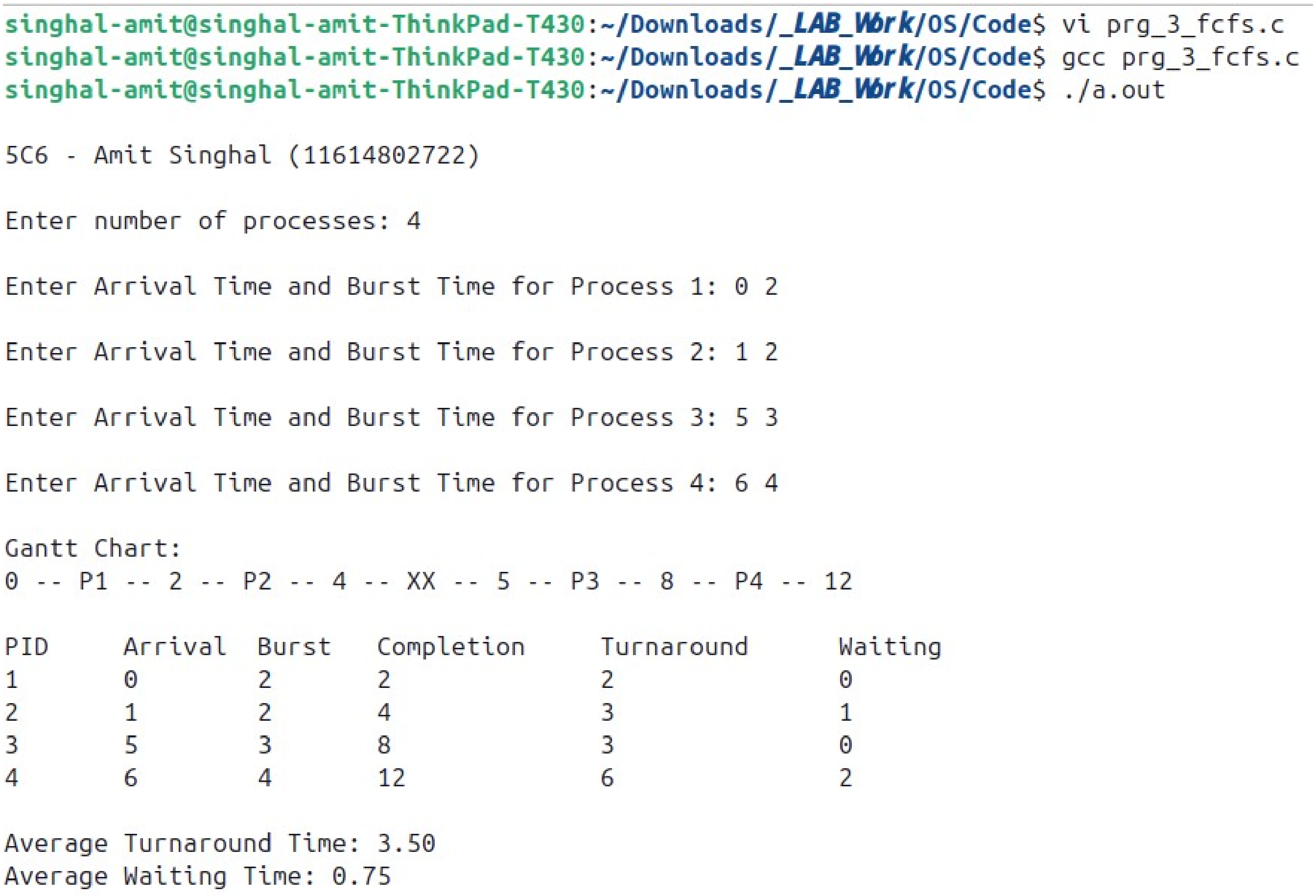
- AIM :: WAP in C to implement CPU scheduling for `first come first serve` (fcfs).

Source_Code ::

```
#include <stdio.h>


typedef struct
{
  int pid;        // Process ID
  int arrival;    // Arrival time
  int burst;      // Burst time
  int completion; // Completion time
  int waiting;    // Waiting time
  int turnaround; // Turnaround time
} Process;


// Function to sort processes by arrival time
void sortByArrival(Process *p, int n)
{
  for (int i = 0; i < n - 1; i++)
  {
    for (int j = 0; j < n - i - 1; j++)
    {
      if (p[j].arrival > p[j + 1].arrival)
      {
        Process temp = p[j];
        p[j] = p[j + 1];
        p[j + 1] = temp;
      }
    }
  }
}

// Main FCFS logic
void fcfsScheduling(Process *p, int n)
{
  int time = 0;

  for (int i = 0; i < n; i++)
  {
    if (time < p[i].arrival)
      time = p[i].arrival; // Set time to the process arrival time if idle

    time += p[i].burst;
    p[i].completion = time;
    p[i].turnaround = p[i].completion - p[i].arrival;
    p[i].waiting = p[i].turnaround - p[i].burst;
  }
}

// Function to display the Gantt chart with idle times
void displayGanttChart(Process *p, int n)
{
  int currentTime = p[0].arrival; // Start from the first process arrival time
  printf("Gantt Chart:\n");

  // Print initial time
  printf("%d", currentTime);

  for (int i = 0; i < n; i++)
  {
    if (currentTime < p[i].arrival)
    {
      // Display idle time
      printf(" -- XX -- %d", p[i].arrival);
      currentTime = p[i].arrival; // Update current time to the arrival of the next process
    }
    // Display the process and its completion time
    printf(" -- P%d -- %d", p[i].pid, p[i].completion);
    currentTime = p[i].completion; // Update current time to the completion of the current process
  }

  printf("\n\n");
}

// Function to calculate and display average times
void calculateAverages(Process *p, int n)
{
  float totalTurnaround = 0, totalWaiting = 0;

  for (int i = 0; i < n; i++)
  {
    totalTurnaround += p[i].turnaround;
    totalWaiting += p[i].waiting;
  }

  printf("\nAverage Turnaround Time: %.2f\n", totalTurnaround / n);
  printf("Average Waiting Time: %.2f\n", totalWaiting / n);
}

// Function to display process information
void displayResults(Process *p, int n) {
  printf("PID\tArrival\t Burst\t Completion\tTurnaround\tWaiting\n");
  for (int i = 0; i < n; i++) {
    printf("%d\t%d\t %d\t %d\t\t%d\t\t%d\n", p[i].pid, p[i].arrival, p[i].burst,
           p[i].completion, p[i].turnaround, p[i].waiting);
  }
}

int main() {
  int n;
  printf("\n5C6 - Amit Singhal (11614802722)\n");
  printf("\nEnter number of processes: ");
  scanf("%d", &n);
  Process p[n];

  for (int i = 0; i < n; i++) {
    printf("\nEnter Arrival Time and Burst Time for Process %d: ", i + 1);
    p[i].pid = i + 1;
    scanf("%d%d", &p[i].arrival, &p[i].burst);
```

```
        p[i].completion = 0; // Initially, no process is completed
    }
    printf("\n");

    sortByArrival(p, n);
    fcfsScheduling(p, n);
    displayGanttChart(p, n);
    displayResults(p, n);
    calculateAverages(p, n);

    printf("\n");

    return 0;
}
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ vi prg_3_fcfs.c
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ gcc prg_3_fcfs.c
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ ./a.out

5C6 - Amit Singhal (11614802722)

Enter number of processes: 4

Enter Arrival Time and Burst Time for Process 1: 0 2

Enter Arrival Time and Burst Time for Process 2: 1 2

Enter Arrival Time and Burst Time for Process 3: 5 3

Enter Arrival Time and Burst Time for Process 4: 6 4

Gantt Chart:
0 -- P1 -- 2 -- P2 -- 4 -- XX -- 5 -- P3 -- 8 -- P4 -- 12

PID     Arrival  Burst   Completion     Turnaround      Waiting
1       0        2       2              2               0
2       1        2       4              3               1
3       5        3       8              3               0
4       6        4       12             6               2

Average Turnaround Time: 3.50
Average Waiting Time: 0.75
```

Lab Exercise – 4

❖ AIM :: WAP in C to implement CPU scheduling for `shortest job first` (sjf).

Source_Code ::

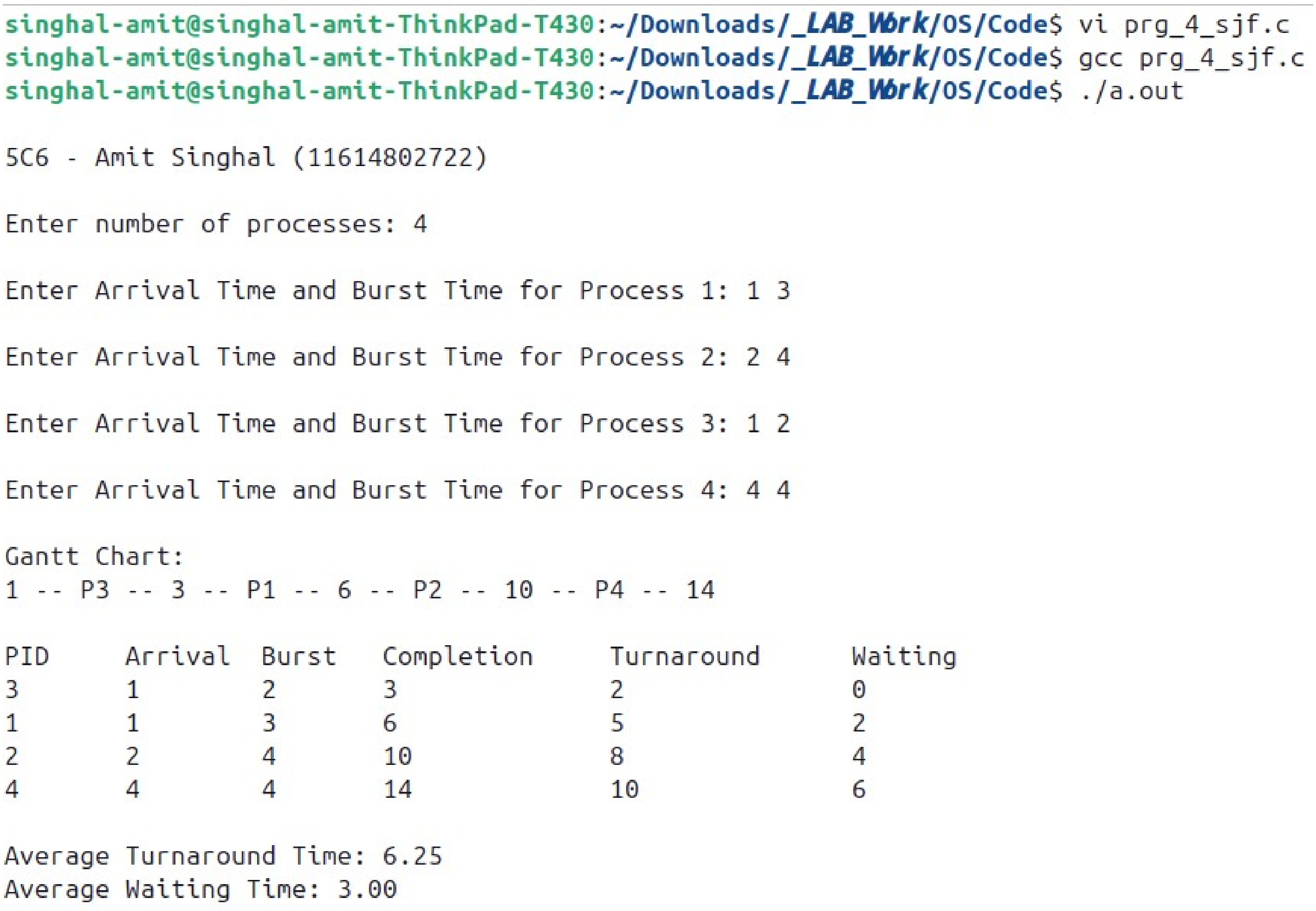
```
#include <stdio.h>


typedef struct
{
  int pid;        // Process ID
  int arrival;    // Arrival time
  int burst;      // Burst time
  int completion; // Completion time
  int waiting;    // Waiting time
  int turnaround; // Turnaround time
} Process;


// Function to sort processes by arrival time, and by burst time in case of tie
void sortByArrival(Process *p, int n)
{
  for (int i = 0; i < n - 1; i++)
  {
    for (int j = 0; j < n - i - 1; j++)
    {
      if (p[j].arrival > p[j + 1].arrival ||
          (p[j].arrival == p[j + 1].arrival && p[j].burst > p[j + 1].burst))
      {
        Process temp = p[j];
        p[j] = p[j + 1];
        p[j + 1] = temp;
      }
    }
  }
}

// Main SJF logic
void sjfScheduling(Process *p, int n)
{
  int time = 0, completed = 0, minIndex;

  while (completed < n)
  {
    minIndex = -1;

    // Find process with min burst time from the pool of arrived processes
    for (int i = 0; i < n; i++)
    {
      if (p[i].arrival <= time && p[i].completion == 0)
      {
        if (minIndex == -1 || p[i].burst < p[minIndex].burst)
    {
     minIndex = i;
    }
   }
  }

  if (minIndex != -1)
  {
   if (time < p[minIndex].arrival)
    time = p[minIndex].arrival; // Set time to the process arrival time if idle

   time += p[minIndex].burst;
   p[minIndex].completion = time;
   p[minIndex].turnaround = p[minIndex].completion - p[minIndex].arrival;
   p[minIndex].waiting = p[minIndex].turnaround - p[minIndex].burst;
   completed++;
  }
  else
  {
   time++;
  }
 }
}

// Function to display the Gantt chart
void displayGanttChart(Process *p, int n)
```

```
{
  int startTime = p[0].arrival;
  printf("Gantt Chart:\n%d", startTime);

  for (int i = 0; i < n; i++)
  {
    printf(" -- P%d -- %d", p[i].pid, p[i].completion);
  }

  printf("\n\n");
}

// Function to calculate and display average times
void calculateAverages(Process *p, int n)
{
  float totalTurnaround = 0, totalWaiting = 0;

  for (int i = 0; i < n; i++)
  {
    totalTurnaround += p[i].turnaround;
    totalWaiting += p[i].waiting;
  }

  printf("\nAverage Turnaround Time: %.2f\n", totalTurnaround / n);
  printf("Average Waiting Time: %.2f\n", totalWaiting / n);
}
```

```
// Function to display process information
void displayResults(Process *p, int n)
{
  printf("PID\tArrival\t Burst\t Completion\tTurnaround\tWaiting\n");
  for (int i = 0; i < n; i++)
  {
    printf("%d\t%d\t %d\t %d\t\t%d\t\t%d\n", p[i].pid, p[i].arrival, p[i].burst,
           p[i].completion, p[i].turnaround, p[i].waiting);
  }
}

int main()
{
  int n;
  printf("\n5C6 - Amit Singhal (11614802722)\n");
  printf("\nEnter number of processes: ");
  scanf("%d", &n);
  Process p[n];

  for (int i = 0; i < n; i++) {
    printf("\nEnter Arrival Time and Burst Time for Process %d: ", i + 1);
    p[i].pid = i + 1;
    scanf("%d%d", &p[i].arrival, &p[i].burst);
    p[i].completion = 0; // Initially, no process is completed
  }
```

```
    printf("\n");


    sortByArrival(p, n);

    sjfScheduling(p, n);

    displayGanttChart(p, n);

    displayResults(p, n);

    calculateAverages(p, n);


    printf("\n");


    return 0;

}
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ vi prg_4_sjf.c
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ gcc prg_4_sjf.c
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ ./a.out

5C6 - Amit Singhal (11614802722)

Enter number of processes: 4

Enter Arrival Time and Burst Time for Process 1: 1 3

Enter Arrival Time and Burst Time for Process 2: 2 4

Enter Arrival Time and Burst Time for Process 3: 1 2

Enter Arrival Time and Burst Time for Process 4: 4 4

Gantt Chart:
1 -- P3 -- 3 -- P1 -- 6 -- P2 -- 10 -- P4 -- 14

PID     Arrival  Burst   Completion     Turnaround      Waiting
3       1        2       3              2               0
1       1        3       6              5               2
2       2        4       10             8               4
4       4        4       14             10              6

Average Turnaround Time: 6.25
Average Waiting Time: 3.00
```

Lab Exercise – 5

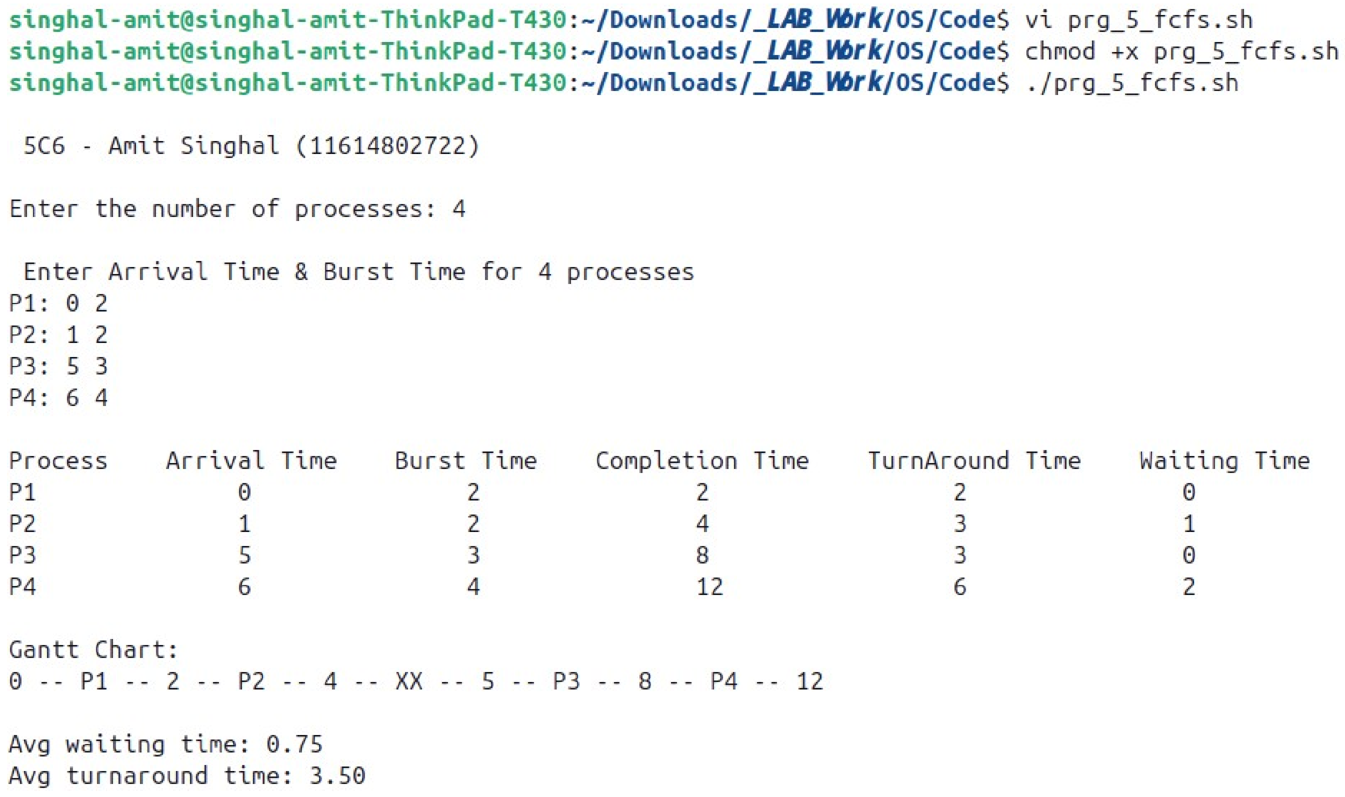
- AIM :: WAP in shell script to implement CPU scheduling for `first come first serve` (fcfs).

Source_Code ::

```
echo $'\n' "5C6 - Amit Singhal (11614802722)" $'\n'

read -p "Enter the number of processes: " num_processes
echo $'\n' "Enter Arrival Time & Burst Time for $num_processes processes"

# Collect process details
for ((i=0;i<num_processes;i++)); do
    echo -n "P$((i+1)): "
    read arrival_time burst_time
    processes[$i]="$arrival_time $burst_time"
done

# Sort processes by arrival time
IFS=$'\n' sorted_processes=($(sort -n -k1 <<<"${processes[*]}"))
unset IFS

# Initialize variables
total_completion_time=0
total_waiting_time=0
total_turnaround_time=0
gantt_chart="0"  # Start Gantt chart at time 0


# Display table header
```

```
echo -e "\nProcess    Arrival Time    Burst Time    Completion Time    TurnAround Time    Waiting Time"

# Process all processes
for ((i=0;i<num_processes;i++)); do
    current_process=(${sorted_processes[$i]})
    current_arrival_time=${current_process[0]}
    current_burst_time=${current_process[1]}

    # If the process arrives after the last completion time, idle CPU
    if (( total_completion_time < current_arrival_time )); then
        idle_time=$((current_arrival_time - total_completion_time))
        total_completion_time=$current_arrival_time
        gantt_chart+=" -- XX -- $total_completion_time"
    fi

    # Calculate waiting time
    if (( total_completion_time >= current_arrival_time )); then
        waiting_time=$((total_completion_time - current_arrival_time))
    else
        waiting_time=0
    fi

    # Calculate completion time and turnaround time
    completion_time=$((total_completion_time + current_burst_time))
    turnaround_time=$((completion_time - current_arrival_time))

    # Update total values
    total_completion_time=$completion_time
    total_waiting_time=$((total_waiting_time + waiting_time))
    total_turnaround_time=$((total_turnaround_time + turnaround_time))

    # Display process details
    echo -e "P$((i+1))\t\t$current_arrival_time\t\t$current_burst_time\t\t$completion_time\t\t  $turnaround_time\t\t  $waiting_time"
    # Update Gantt chart
    gantt_chart+=" -- P$((i+1)) -- $completion_time"
done

# Calculate averages
avg_waiting_time=$(awk "BEGIN {printf \"%.2f\", $total_waiting_time/$num_processes}")
avg_turnaround_time=$(awk "BEGIN {printf \"%.2f\", $total_turnaround_time/$num_processes}")

# Display Gantt chart
echo -e "\nGantt Chart:"
echo -e "$gantt_chart"

# Display averages
echo ""
echo "Avg waiting time: $avg_waiting_time"
echo "Avg turnaround time: $avg_turnaround_time"
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ vi prg_5_fcfs.sh
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ chmod +x prg_5_fcfs.sh
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ ./prg_5_fcfs.sh

 5C6 - Amit Singhal (11614802722)

Enter the number of processes: 4

 Enter Arrival Time & Burst Time for 4 processes
P1: 0 2
P2: 1 2
P3: 5 3
P4: 6 4

Process     Arrival Time     Burst Time     Completion Time     TurnAround Time     Waiting Time
P1                0               2                2                   2                  0
P2                1               2                4                   3                  1
P3                5               3                8                   3                  0
P4                6               4                12                  6                  2

Gantt Chart:
0 -- P1 -- 2 -- P2 -- 4 -- XX -- 5 -- P3 -- 8 -- P4 -- 12

Avg waiting time: 0.75
Avg turnaround time: 3.50
```

Lab Exercise – 6

- AIM :: WAP in shell script to implement CPU scheduling for `shortest job first` (sjf).

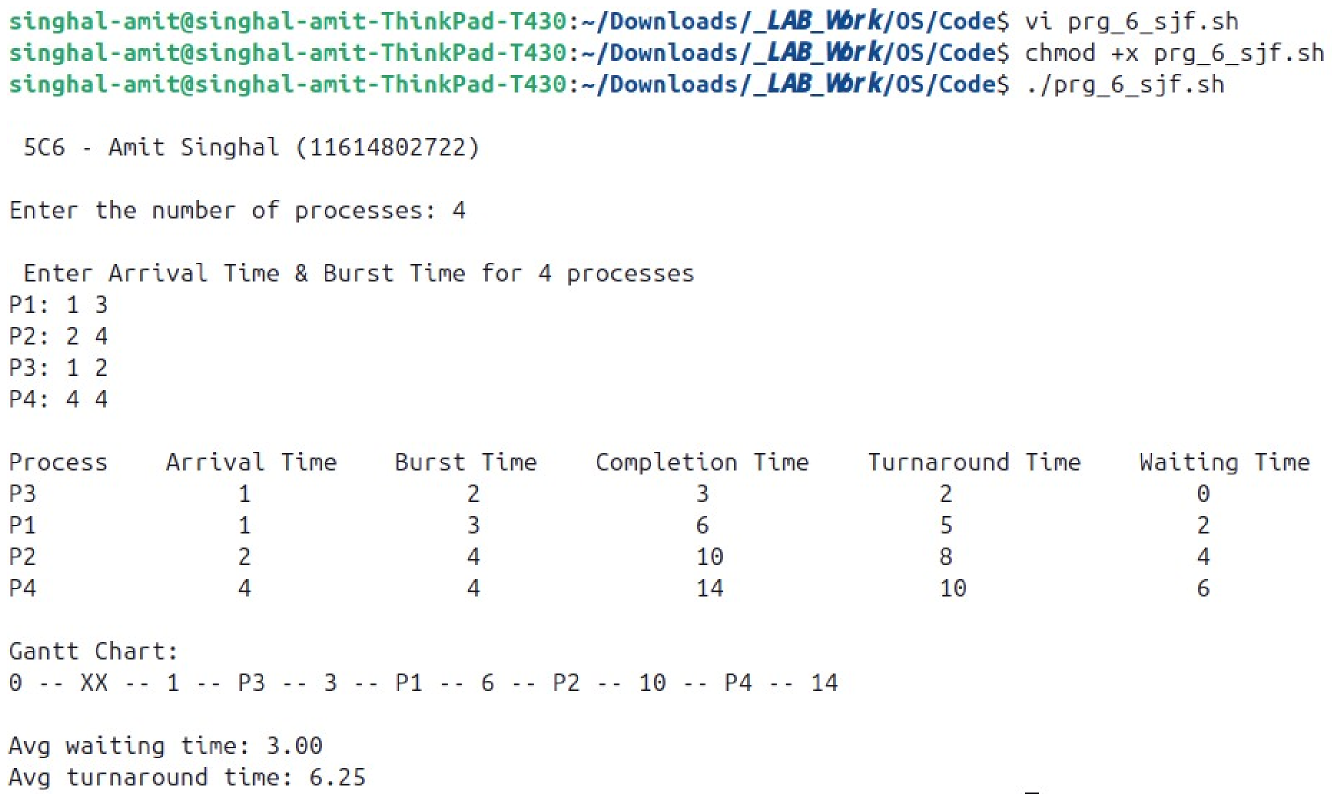
Source_Code ::

```
echo $'\n' "5C6 - Amit Singhal (11614802722)" $'\n'

read -p "Enter the number of processes: " num_processes
echo $'\n' "Enter Arrival Time & Burst Time for $num_processes processes"

# Collect process details
for ((i=0;i<num_processes;i++)); do
    echo -n "P$((i+1)): "
    read arrival_time burst_time
    processes[$i]="$arrival_time $burst_time"
done

# Initialize variables
total_completion_time=0
total_waiting_time=0
total_turnaround_time=0
completed_processes=0
gantt_chart="0"  # Start Gantt chart at time 0
time=0

# Create an array to store completion status of each process (0 = incomplete, 1 = complete)
for ((i=0;i<num_processes;i++)); do
```

```
    process_completed[$i]=0
done

# Function to find the process with the shortest burst time among those that have arrived
find_shortest_job() {
    local min_burst=-1
    local min_index=-1

    for ((i=0;i<num_processes;i++)); do
        current_process=(${processes[$i]})
        current_arrival_time=${current_process[0]}
        current_burst_time=${current_process[1]}

        if (( process_completed[$i] == 0 && current_arrival_time <= time )); then
            if (( min_burst == -1 || current_burst_time < min_burst )); then
                min_burst=$current_burst_time
                min_index=$i
            fi
        fi
    done

    echo $min_index
}

# Display table header
echo -e "\nProcess    Arrival Time    Burst Time    Completion Time    Turnaround Time    Waiting Time"

# Process all processes using SJF
while (( completed_processes < num_processes )); do
    shortest_job=$(find_shortest_job)

    if (( shortest_job == -1 )); then
```

```
        # No process available, increase time (idle)
        gantt_chart+=" -- XX -- $((++time))"
    else
        current_process=(${processes[$shortest_job]})
        current_arrival_time=${current_process[0]}
        current_burst_time=${current_process[1]}

        if (( time < current_arrival_time )); then
            time=$current_arrival_time
            gantt_chart+=" -- XX -- $time"
        fi

        completion_time=$((time + current_burst_time))
        turnaround_time=$((completion_time - current_arrival_time))
        waiting_time=$((turnaround_time - current_burst_time))

        # Update total values
        total_completion_time=$completion_time
        total_waiting_time=$((total_waiting_time + waiting_time))
        total_turnaround_time=$((total_turnaround_time + turnaround_time))

        # Mark the process as completed
        process_completed[$shortest_job]=1
        completed_processes=$((completed_processes + 1))

        # Display process details
        echo -e "P$((shortest_job+1))\t\t$current_arrival_time\t\
t$current_burst_time\t\t$completion_time\t\t $turnaround_time\t\t
$waiting_time"

        # Update Gantt chart
        gantt_chart+=" -- P$((shortest_job+1)) -- $completion_time"

        # Update current time
        time=$completion_time
    fi
  done

  # Calculate averages
  avg_waiting_time=$(awk "BEGIN {printf \"%.2f\",
  $total_waiting_time/$num_processes}")
  avg_turnaround_time=$(awk "BEGIN {printf \"%.2f\",
  $total_turnaround_time/$num_processes}")

  # Display Gantt chart
  echo -e "\nGantt Chart:"
  echo -e "$gantt_chart"

  # Display averages
  echo ""
  echo "Avg waiting time: $avg_waiting_time"
  echo "Avg turnaround time: $avg_turnaround_time"
```

Output ::



```
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ vi prg_6_sjf.sh
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ chmod +x prg_6_sjf.sh
singhal-amit@singhal-amit-ThinkPad-T430:~/Downloads/_LAB_Work/OS/Code$ ./prg_6_sjf.sh

 5C6 - Amit Singhal (11614802722)

Enter the number of processes: 4

 Enter Arrival Time & Burst Time for 4 processes
P1: 1 3
P2: 2 4
P3: 1 2
P4: 4 4

Process    Arrival Time    Burst Time    Completion Time    Turnaround Time    Waiting Time
P3              1               2               3                2                 0
P1              1               3               6                5                 2
P2              2               4               10               8                 4
P4              4               4               14               10                6

Gantt Chart:
0 -- XX -- 1 -- P3 -- 3 -- P1 -- 6 -- P2 -- 10 -- P4 -- 14

Avg waiting time: 3.00
Avg turnaround time: 6.25
```