1. a. **Write a Shell Program to find the factorial of a given n number.**

Program:

```
echo Code to find Factorial of n numbers

echo Enter the number :

read n

fact=1

for((i=1 ; i<=n ; i++))

do

fact=$(expr $fact \* $i)

done

echo factorial of $n numbers is $fact
```

b. **Write a C/C++ Program to simulate the copy, edit and rename command.**

```
// 1.ii)C++ program for implementation of FCFS
// scheduling
#include<iostream>
using namespace std;

// Function to find the waiting time for all
// processes
void findWaitingTime(int processes[], int n,
                                        int bt[], int wt[])
{
        // waiting time for first process is 0
        wt[0] = 0;

        // calculating waiting time
        for (int i = 1; i < n ; i++ )
                wt[i] = bt[i-1] + wt[i-1] ;
```

```
}

// Function to calculate turn around time
void findTurnAroundTime( int processes[], int n,
                         int bt[], int wt[], int tat[])
{
        // calculating turnaround time by adding
        // bt[i] + wt[i]
        for (int i = 0; i < n ; i++)
                tat[i] = bt[i] + wt[i];
}

//Function to calculate average time
void findavgTime( int processes[], int n, int bt[])
{
        int wt[n], tat[n], total_wt = 0, total_tat = 0;

        //Function to find waiting time of all processes
        findWaitingTime(processes, n, bt, wt);

        //Function to find turn around time for all processes
        findTurnAroundTime(processes, n, bt, wt, tat);

        //Display processes along with all details
        cout << "Processes "<< " Burst time "
                << " Waiting time " << " Turn around time\n";

        // Calculate total waiting time and total turn
        // around time
        for (int i=0; i<n; i++)
        {
                total_wt = total_wt + wt[i];
                total_tat = total_tat + tat[i];
                cout << " " << i+1 << "\t\t" << bt[i] <<"\t "
                        << wt[i] <<"\t\t " << tat[i] <<endl;
        }

        cout << "Average waiting time = "
                << (float)total_wt / (float)n;
        cout << "\nAverage turn around time = "
```

```
                << (float)total_tat / (float)n;
}

// Driver code
int main()
{
        //process id's
        int processes[] = { 1, 2, 3,4,5};
        int n = sizeof processes / sizeof processes[0];

        //Burst time of all processes
        int burst_time[] = {2,8,4,6,2};

        findavgTime(processes, n, burst_time);
        return 0;
}
```

2. a. **Write a Shell Program to find the Fibonacci of n numbers.**

Program:

```
echo Code to find Fibonacci of n numbers

echo Enter the number :

read n

f1=0

f2=1

echo Fibonacci of $n numbers :

echo $f1

echo $f2

for((i=2 ; i<n ; i++))

do

k=$(expr $f1 \+ $f2)

f1=$f2

f2=$k
```

```
        echo $k
        done
```

b. Write a C/C++ program to simulate FCFS/FIFO CPU scheduling algorithm.

Program:

```
#include<iostream>
using namespace std;
int main() {
   cout<<"Enter number of Process : ";
   int n;
   cin>>n;
   cout<<"Enter the process\n";
   int process[n];
   int burst_time[n];
   for(int i=0 ; i<n ; i++) {
      cout<<"Process P"<<(i+1)<<": ";
      cin>>process[i];
      cout<<"Burst Time : ";
      cin>>burst_time[i];
   }
   int turn_around_time[n], wait_time[n];
   turn_around_time[0] = burst_time[0];
   for(int i=0 ; i<n ; i++) {
      turn_around_time[i] = turn_around_time[i-1]+burst_time[i];
   }
   for(int i=0 ; i<n ; i++) {
```

```
        wait_time[i] = turn_around_time[i]-burst_time[i];
    }
    int tot_wt=0,tot_tat=0;
    for(int i=0 ; i<n ; i++) {
        tot_tat += turn_around_time[i];
        tot_wt += wait_time[i];
    }
    cout<<"Process\tBurst Time\tTurn Around Time\tWait Time\n";
    for(int i=0 ; i<n ; i++) {

cout<<process[i]<<"\t\t"<<burst_time[i]<<"\t\t"<<turn_around_tim
e[i]<<"\t\t"<<wait_time[i]<<"\n";
    }
    cout<<"Average TAT = "<<(float)tot_tat/n;
    cout<<"\nAverage WT = "<<(float)tot_wt/n;
}
```

3. a. Write a shell program to perform file operations.

```
#!/bin/bash

# Creating a file
touch myfile.txt

# Writing to the file
echo "Hello, this is a test file." > myfile.txt

# Reading from the file
cat myfile.txt

# Deleting the file
rm myfile.txt
```

b. Write a C/C++ program to simulate FCFS/FIFO Disk scheduling algorithm.

Program:

```
#include<bits/stdc++.h>
using namespace std;
int main() {
   int n;
   cout<<"Enter the size of the queue : ";
   cin>>n;
   cout<<"Enter the disk queue : ";
   int disk[n];
   for(int i=0 ; i<n ; i++) {
      cin>>disk[i];
   }
   cout<<"Enter the initial head position : ";
   int head;
   cin>>head;
   int tot_head_time = 0, no_of_head_movements = 0;
   for(int i=0 ; i<n ; i++) {
      tot_head_time += abs(head-disk[i]);
      head = disk[i];
      no_of_head_movements++;
   }
   cout<<"Toal seek time : "<<tot_head_time;
   cout<<"\nNumber of head movements :
"<<no_of_head_movements;
```

}

4. a. Write a Shell Program to find the sum of n numbers

Program:

```
echo Code to find the Sum of n numbers
echo Enter the number :
read n
sum=0
for((i=1 ; i<=n ; i++))
do
sum=$(expr $sum \+ $i)
done
echo Sum of $n numbers is $sum
```

b. Write a C/C++ program to simulate producer consumer problem.

5. a. Write a shell program to find the greatest of three numbers.

Program:

```
echo Code to find the greatest of 3 numbers
echo Enter the numbers
 read a b c
 if [ $a -ge $b ] && [ $a -ge $c ]
 then
 echo The greatest of given 3 numbers is a : $a
 elif [ $b -gt $c ]
 then
 echo The greatest of given 3 numbers is b : $b
 else
```

```
        echo The greatest of given 3 numbers is c : $c
        fi
```

b. **Write a C/C++ program to simulate FCFS page replacement algorithm.**

Program:

```
//fifo PAGE REPLACEMENT
#include<bits/stdc++.h>
using namespace std;
int main()
{
   //No of inputs in sequence
   int n;
   cin>>n;
   //The Sequence
   int arr[n];
   for(int i=0; i<n; i++)
   {
      cin>>arr[i];
   }
   //The number of frames
   int m;
   cin>>m;
   map<int,int> mpp;
   for(int i=1; i<=m; i++)
      mpp[i] = -1;
   queue<int> q;
```

```
set<int> st;
int cnt = 0;
for(int i=0; i<n; i++)
{
   if(st.find(arr[i])!=st.end())
   {
      for(int frame=1; frame<=m; frame++)
         cout<<mpp[frame]<<" ";
   }else{
      if(q.size() == m)
      {
         int numberToBeRemoved = q.front();
         st.erase(q.front());
         q.pop();
         int ind = 0;
         for(int frame=1; frame<=m; frame++)
         {
            if(mpp[frame] == numberToBeRemoved)
            {
               ind = frame;
               break;
            }
         }
         mpp[ind] = arr[i];
         q.push(arr[i]);
         st.insert(arr[i]);
```

```
            }else{
               q.push(arr[i]);
               st.insert(arr[i]);
               mpp[q.size()] = arr[i];
            }
            cnt++;
            for(int frame = 1;frame<=m; frame++)
            {
               cout<<mpp[frame]<<" ";
            }
         }
         cout<<endl;
      }
      cout<<"Page Faults : "<<cnt<<endl;
   }
```

6. a. Write a C program to check whether a given file is in a directory or not.

   b. Write a C/C++ program to simulate SJF CPU scheduling algorithm.

   Program:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   cout<<"SJF CPU Scheduling\n";
   cout<<"Enter the number of Process : ";
   int n;
```

```
cin>>n;
pair<int,int> p[n];
for(int i=0; i<n; i++)
{
   cout<<"Burst time of Process P"<<(i+1)<<" : ";
   cin>>p[i].first;
   p[i].second = i;
}
int wait[n],tat[n];
sort(p,p+n);
int crntTime = 0;
for(int i=0; i<n; i++)
{
   int ind = p[i].second;
   wait[ind] = crntTime;
   crntTime += p[i].first;
   tat[ind] = crntTime;
}
cout<<"process waitTime\ttat\n";
double tot_wt = 0,tot_tat = 0;
for(int i=0; i<n; i++)
{
   cout<<i + 1<<"\t\t"<<wait[i]<<"\t\t"<<tat[i]<<endl;
   tot_wt += wait[i];
   tot_tat += tat[i];
}
```

```
    cout<<"avg Wait Time\tavg Tat\n";
    cout<<tot_wt/(double)n<<"\t\t"<<tot_tat/(double)n<<endl;
  }
```

7. a. Write a C program simulate process system calls.

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

int main() {
  // Create a child process
  int pid = fork();

  if (pid > 0) {
    printf("I am parent process:\n");
    printf("PID: %d\n", getpid());
    printf("Child's PID: %d\n", pid);
  }
  else if (pid == 0) {
    printf("\nI am child process:\n");
    printf("PID: %d\n", getpid());
    printf("Parent's PID: %d\n", getppid());
  }
  else {
    printf("Failed to create child process.\n");
  }

  return 0;
}
```

b. Write a C/C++ program to check whether a given system is safe or not.

8. a. Write a shell program to find the sum of n numbers.

Program:

```
echo Code to find the Sum of n numbers
echo Enter the number :
```

```
read n
sum=0
for((i=1 ; i<=n ; i++))
do
sum=$(expr $sum \+ $i)
done
echo Sum of $n numbers is $sum
```

b. Write a C/C++ program to simulate priority CPU scheduling algorithms.

Program:

```
#include<bits/stdc++.h>
using namespace std;
bool comp(pair<int,pair<int,int>> &a,pair<int,pair<int,int>> &b)
{
   if(a.first == b.first)
   {
      return a.second.second< b.second.second;
   }
   return a.first<b.first;
}
int main()
{
   cout<<"Enter the number of Process : ";
   int n;
   cin>>n;
   pair<int,pair<int,int>> p[n];
```

```
for(int i=0; i<n; i++)
{
   cout<<"Priority of Process P"<<(i+1)<<": ";
   cin>>p[i].first;
   cout<<"Burst time of Process P"<<(i+1)<<" : ";
   cin>>p[i].second.first;
   p[i].second.second = i;
}
int wait[n],tat[n];
sort(p,p+n,comp);
int crntTime = 0;
for(int i=0; i<n; i++)
{
   int ind = p[i].second.second;
   wait[ind] = crntTime;
   crntTime += p[i].second.first;
   tat[ind] = crntTime;
}
cout<<"process waitTime\ttat\n";
double tot_wt = 0,tot_tat = 0;
for(int i=0; i<n; i++)
{
   cout<<i + 1<<"\t\t"<<wait[i]<<"\t\t"<<tat[i]<<endl;
   tot_wt += wait[i];
   tot_tat += tat[i];
}
```

```
    cout<<"avg Wait Time\tavg Tat\n";
    cout<<tot_wt/(double)n<<"\t\t"<<tot_tat/(double)n<<endl;
}
```

9. a. **Write a menu driven Shell Programming to perform the following**

i)To check whether a given year is leap year or not.

ii)To find whether a given number is positive or negative or neither.

Program:

```
echo Menu driven Shell Program
echo Choice 1 : To find the given year is leap year or not
echo Choice 2 : To find given number is positive, negative or neither
echo Enter your Choice
read choice
case $choice in
1)echo Enter the year :
read year
b=$(expr $year \% 4)
if [ $b -eq 0 ]
then
echo $year is a Leap year
else
echo $year is not a Leap year
fi
;;
2)echo Enter the number :
read n
```

```
if [ $n -gt 0 ]
then
echo The number $n is positive
elif [ $n -lt 0 ]
then
echo The number $n is negative
else
echo The number $n is neither positive nor negative
fi
 ;;
*)echo Enter the correct choice
;;
esac
```

b. Write a C/C++ program to perform
   i) stat system calls
   ii) FIFO Disk scheduling algorithm.
   Program:

```
#include<bits/stdc++.h>
using namespace std;
int main() {
   int n;
   cout<<"Enter the size of the queue : ";
   cin>>n;
   cout<<"Enter the disk queue : ";
   int disk[n];
   for(int i=0 ; i<n ; i++) {
      cin>>disk[i];
```

```
    }
    cout<<"Enter the initial head position : ";
    int head;
    cin>>head;
    int tot_head_time = 0, no_of_head_movements = 0;
    for(int i=0 ; i<n ; i++) {
      tot_head_time += abs(head-disk[i]);
      head = disk[i];
      no_of_head_movements++;
    }
    cout<<"Toal seek time : "<<tot_head_time;
    cout<<"\nNumber of head movements :
"<<no_of_head_movements;
    }
```

10. a. Write a shell program to find whether the given number is odd or even.

Program:

```
echo Program to find whether the given number is odd or
even
echo Enter the number :
read n
rem=$(expr $n \% 2)
if [ $rem -eq 0 ]
then
echo The number $n is even
else
echo The number $n is odd
fi
```

b. Write c/c++ program

i)to simulate FIFO page replacement algorithm.

Program:

```
//fifo PAGE REPLACEMENT
```

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   //No of inputs in sequence
   int n;
   cin>>n;
   //The Sequence
   int arr[n];
   for(int i=0; i<n; i++)
   {
      cin>>arr[i];
   }
   //The number of frames
   int m;
   cin>>m;
   map<int,int> mpp;
   for(int i=1; i<=m; i++)
      mpp[i] = -1;
   queue<int> q;
   set<int> st;
   int cnt = 0;
```

```
for(int i=0; i<n; i++)
{
   if(st.find(arr[i])!=st.end())
   {
      for(int frame=1; frame<=m; frame++)
         cout<<mpp[frame]<<" ";
   }else{
      if(q.size() == m)
      {
         int numberToBeRemoved = q.front();
         st.erase(q.front());
         q.pop();
         int ind = 0;
         for(int frame=1; frame<=m; frame++)
         {
            if(mpp[frame] == numberToBeRemoved)
            {
               ind = frame;
               break;
            }
         }
         mpp[ind] = arr[i];
```

```
            q.push(arr[i]);
            st.insert(arr[i]);
        }else{
            q.push(arr[i]);
            st.insert(arr[i]);
            mpp[q.size()] = arr[i];
        }
        cnt++;
        for(int frame = 1;frame<=m; frame++)
        {
            cout<<mpp[frame]<<" ";
        }
      }
      cout<<endl;
   }
   cout<<"Page Faults : "<<cnt<<endl;
}
```

ii)perform operations on fork() and exec() system calls.