

**Ramakrishna Mission Vidyamandira**

(An Autonomous College under University of Calcutta)

Computer Science (Honors) Semester III 2022

Paper: CMSA CC 6 Practical

|  |
| --- |
| **Krishnendu Das** |
| Class Roll Number: 302  Registration Number: A04-1112-0173-21  B.Sc.  3rd Semester  Batch: 2021-24 |

***ASSIGNMENT 1***

*#Write a shell program to print average marks and grade of students upon three input marks.*

*#Submitted by Roll No: 302*

*#Source Code*

*#!bin/bash/*

echo "Enter the marks of Physics:"

read n

echo "Enter the marks of Chemistry:"

read m

echo "Enter the marks of Maths:"

read o

sum=`expr $n + $m + $o`

avg=`expr $sum / 3`

echo "You Got $sum"

if [ $avg -ge 80 ]

then

    echo "Calculating Grade..."

    sleep 2

    echo "Your Grade=Letter"

elif [ $avg -ge 60 ]

then

    echo "Calculating Grade..."

    sleep 2

    echo "Your Grade=First"

elif [ $avg -ge 40 ]

then

    echo "Calculating Grade..."

    sleep 2

    echo "Your Grade=Second"

else

    echo "Calculating Grade..."

    sleep 2

    echo "Your Grade=Fail"

fi

*###############OUTPUT#############*

*# Enter the marks of Physics:*

*# 80*

*# Enter the marks of Chemistry:*

*# 85*

*# Enter the marks of Maths:*

*# 75*

*# You Got 240*

*# Calculating Grade...*

*# Your Grade=Letter*

*#Write a shell program to print sum of digits of a three-digit number.*

*#Submitted by Roll No: 302*

*#Source Code*

echo "Enter a number"

read num

sum=0

while [ $num -gt 0 ]

do

    mod=`expr $num % 10`

    sum=`expr $sum + $mod`

    num=`expr $num / 10`

done

echo $sum

*###################output##############*

*# Enter a number*

*# 586*

*# 19*

*# Enter a number*

*# 536436*

*# 27*

*#Write a shell program to check whether a number is odd or even.*

*#Submitted by Roll No: 302*

*#Source Code*

*#!bin/bash*

echo "Enter an integer"

read a

var=`expr $a % 2`

if [ $var -eq 0 ]

then

    echo " $a is even"

else

    echo " $a is odd"

fi

*#############output##########*

*# Enter an integer*

*# 463*

*# 463 is odd*

*# Enter an integer*

*# 24*

*# 24 is even*

*#Write a shell program to print factorial of a number.*

*#Submitted by Roll No: 302*

*#Source Code*

echo "Enter a number"

read num

fact=1

while [ $num -gt 1 ]

do

  fact=$((fact \* num))

  num=$((num - 1))

done

echo $fact

*############OUTPUT############*

*# Enter a number*

*# 7*

*# 5040*

*#Write a shell program to print Fibonacci series.*

*#Submitted by Roll No:302*

*#Source Code*

echo "Enter The Value of N :"

read N

echo "The Fibonacci series is :"

a=0

b=1

for (( i=0; i<N; i++ ))

do

    echo "$a "

    fn=$((a + b))

    a=$b

    b=$fn

done

*################OUTPUT###############*

*# Enter The Value of N :*

*# 6*

*# The Fibonacci series is :*

*# 0*

*# 1*

*# 1*

*# 2*

*# 3*

*# 5*

*#Write a shell program to print average of first n natural numbers.*

*#Submitted by Roll No : 302*

*#Source Code*

echo "Enter Size(N)"

read N

i=1

sum=0

echo "Enter Numbers"

while [ $i -le $N ]

do

  read num

  sum=$((sum + num))

  i=$((i + 1))

done

avg=$(echo $sum / $N | bc -l)

echo $avg

*#######################OUTPUT################*

*# Enter Size(N)*

*# 4*

*# Enter Numbers*

*# 1*

*# 2*

*# 3*

*# 4*

*# 2.50000000000000000000*

*#Write a shell program to print prime numbers in a given range. Take inputs as command line arguments.*

*#Submitted by Roll No :302*

*#Source Code*

*#!/bin/bash*

prime\_1=0

echo "enter the range"

read n

echo " Prime number between 1 to $n is:"

echo "1"

echo "2"

for((i=3;i<=n;))

do

  for((j=i-1;j>=2;))

  do

    if [  `expr $i % $j` -ne 0 ] ; then

      prime\_1=1

    else

      prime\_1=0

      break

    fi

    j=`expr $j - 1`

  done

  if [ $prime\_1 -eq 1 ] ; then

    echo $i

  fi

  i=`expr $i + 1`

done

*#######################OUTPUT####################*

*# enter the range*

*# 6*

*# Prime number between 1 to 6 is:*

*# 1*

*# 2*

*# 3*

*# 5*

*#Write a shell program to print access permission of a file. Filename should be provided as command line input.*

*#Submitted by Roll No : 302*

*#Source Code*

*#!bin/bash*

echo "The file name is" :$1

if [ -e $1 ]

then

    echo "File Exist"

    if [ -r $1 ]

    then

        echo "Read Permitted"

    else

        echo "Not Readable"

    fi

    if [ -w $1 ]

    then

        echo "Write Permitted"

    else

        echo "Write NOT Permitted"

    fi

    if [ -x $1 ]

    then

        echo "Executable"

    else

        echo "Non executable"

    fi

else

    echo "File does not exist"

fi

*#######################OUTPUT####################*

*# The file name is :fcfs.c*

*# File Exist*

*# Read Permitted*

*# Write Permitted*

*# Executable*

*//Write a program in C to demonstrate First Come First Serve (FCFS) scheduling algorithm and print the waiting times for each process and also print the average waiting time.*

*//Submitted by Roll No: 302*

*//Source Code*

#include <stdio.h>

int main()

{

    int bt[10] = {0}, at[10] = {0}, tat[10] = {0}, wt[10] = {0}, ct[10] = {0};

    int n, sum = 0;

    float totalTAT = 0, totalWT = 0;

    printf("Enter number of processes   ");

    scanf("%d", &n);

    printf("Enter arrival time and burst time for each process\n\n");

    for (int i = 0; i < n; i++)

    {

        printf("Arrival time of process[%d] ", i + 1);

        scanf("%d", &at[i]);

        printf("Burst time of process[%d]   ", i + 1);

        scanf("%d", &bt[i]);

        printf("\n");

    }

*// calculate completion time of processes*

    for (int j = 0; j < n; j++)

    {

        sum += bt[j];

        ct[j] += sum;

    }

*// calculate turnaround time and waiting times*

    for (int k = 0; k < n; k++)

    {

        tat[k] = ct[k] - at[k];

        totalTAT += tat[k];

    }

    for (int k = 0; k < n; k++)

    {

        wt[k] = tat[k] - bt[k];

        totalWT += wt[k];

    }

    printf("Solution: \n\n");

    printf("P#\t AT\t BT\t CT\t TAT\t WT\t\n\n");

    for (int i = 0; i < n; i++)

    {

        printf("P%d\t %d\t %d\t %d\t %d\t %d\n", i + 1, at[i], bt[i], ct[i], tat[i], wt[i]);

    }

    printf("\n\nAverage Turnaround Time = %f\n", totalTAT / n);

    printf("Average WT = %f\n\n", totalWT / n);

    return 0;

}

*/////////////////////////////OUTPUT/////////////////////////////////////*

*// Enter number of processes       4*

*// Enter arrival time and burst time for each process*

*// Arrival time of process[1]      0*

*// Burst time of process[1]        4*

*// Arrival time of process[2]      2*

*// Burst time of process[2]        7*

*// Arrival time of process[3]      3*

*// Burst time of process[3]        8*

*// Arrival time of process[4]      5*

*// Burst time of process[4]        9*

*// Solution:*

*// P#       AT      BT      CT      TAT     WT*

*// P1       0       4       4       4       0*

*// P2       2       7       11      9       2*

*// P3       3       8       19      16      8*

*// P4       5       9       28      23      14*

*// Average Turnaround Time = 13.000000*

*// Average WT = 6.000000*

*//Write a program in C to demonstrate Shortest Job First (SJF) scheduling algorithm and print the waiting times for each process and also print the average waiting time.*

*//Submitted by Roll No: 302*

*//Source Code*

#include <stdio.h>

int main()

{

    int bt[20], p[20], wt[20], tat[20], i, j, n, total = 0, pos, temp;

    float avg\_wt, avg\_tat;

    printf("Enter number of process:");

    scanf("%d", &n);

    printf("\nEnter Burst Time:\n");

    for (i = 0; i < n; i++)

    {

        printf("p%d:", i + 1);

        scanf("%d", &bt[i]);

        p[i] = i + 1;

    }

    for (i = 0; i < n; i++)

    {

        pos = i;

        for (j = i + 1; j < n; j++)

        {

            if (bt[j] < bt[pos])

                pos = j;

        }

        temp = bt[i];

        bt[i] = bt[pos];

        bt[pos] = temp;

        temp = p[i];

        p[i] = p[pos];

        p[pos] = temp;

    }

    wt[0] = 0;

    for (i = 1; i < n; i++)

    {

        wt[i] = 0;

        for (j = 0; j < i; j++)

            wt[i] += bt[j];

        total += wt[i];

    }

    avg\_wt = (float)total / n;

    total = 0;

    printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");

    for (i = 0; i < n; i++)

    {

        tat[i] = bt[i] + wt[i];

        total += tat[i];

        printf("\np%d\t\t  %d\t\t    %d\t\t\t%d", p[i], bt[i], wt[i], tat[i]);

    }

    avg\_tat = (float)total / n;

    printf("\n\nAverage Waiting Time=%f", avg\_wt);

    printf("\nAverage Turnaround Time=%f\n", avg\_tat);

}

*///////////////////////////OUTPUT///////////////////////////////*

*// Enter number of process:4*

*// Enter Burst Time:*

*// p1:4*

*// p2:7*

*// p3:9*

*// p4:5*

*// Process     Burst Time          Waiting Time    Turnaround Time*

*// p1                4                 0                   4*

*// p4                5                 4                   9*

*// p2                7                 9                   16*

*// p3                9                 16                  25*

*// Average Waiting Time=7.250000*

*// Average Turnaround Time=13.500000*