

**Ramakrishna Mission Vidyamandira**

(An Autonomous College under University of Calcutta)

Computer Science (Honors) Semester III 2022

Paper: CMSA CC 6 Practical

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

# Program-10

*// 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 : 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//////////////////////*

*// PS C:\Users\Krishnendu Das\OneDrive\Desktop\Shell and OS> gcc fcfs.c*

*// PS C:\Users\Krishnendu Das\OneDrive\Desktop\Shell and OS> ./a.exe*

*// Enter number of processes:       5*

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

*// Arrival time of process[1]      3*

*// Burst time of process[1]        5*

*// Arrival time of process[2]      3*

*// Burst time of process[2]        6*

*// Arrival time of process[3]      4*

*// Burst time of process[3]        6*

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

*// Burst time of process[4]        7*

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

*// Burst time of process[5]        7*

*// Solution:*

*// P#       AT      BT      CT      TAT     WT*

*// P1       3       5       5       2       -3*

*// P2       3       6       11      8       2*

*// P3       4       6       17      13      7*

*// P4       5       7       24      19      12*

*// P5       4       7       31      27      20*

*// Average Turnaround Time = 13.800000*

*// Average WT = 7.600000*

*// PS C:\Users\Krishnendu Das\OneDrive\Desktop\Shell and OS>*

# Program-11

*//  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;

    }

*//sorting of burst times*

    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////////////////*

*// PS C:\Users\Krishnendu Das\OneDrive\Desktop\Shell and OS> ./a.exe*

*// Enter number of process:5*

*// Enter Burst Time:*

*// p1:5*

*// p2:6*

*// p3:7*

*// p4:4*

*// p5:3*

*// Process     Burst Time          Waiting Time    Turnaround Time*

*// p5                3                 0                   3*

*// p4                4                 3                   7*

*// p1                5                 7                   12*

*// p2                6                 12                  18*

*// p3                7                 18                  25*

*// Average Waiting Time=8.000000*

*// Average Turnaround Time=13.000000*

*// PS C:\Users\Krishnendu Das\OneDrive\Desktop\Shell and OS>*

# Program-12

*// Write a program in C to demonstrate Round Robin (RR) 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>

#include <conio.h>

void main()

{

    int i, NOP, sum = 0, count = 0, y, quant, wt = 0, tat = 0, at[10], bt[10], temp[10];

    float avg\_wt, avg\_tat;

    printf(" Total number of process in the system: ");

    scanf("%d", &NOP);

    y = NOP;

    for (i = 0; i < NOP; i++)

    {

        printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i + 1);

        printf(" Arrival time is: \t");

        scanf("%d", &at[i]);

        printf(" \nBurst time is: \t");

        scanf("%d", &bt[i]);

        temp[i] = bt[i];

    }

    printf("Enter the Time Quantum for the process: \t");

    scanf("%d", &quant);

    printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");

    for (sum = 0, i = 0; y != 0;)

    {

        if (temp[i] <= quant && temp[i] > 0)

        {

            sum = sum + temp[i];

            temp[i] = 0;

            count = 1;

        }

        else if (temp[i] > 0)

        {

            temp[i] = temp[i] - quant;

            sum = sum + quant;

        }

        if (temp[i] == 0 && count == 1)

        {

            y--;

            printf("\nProcess No[%d] \t\t %d\t\t\t %d\t\t\t %d", i + 1, bt[i], sum - at[i], sum - at[i] - bt[i]);

            wt = wt + sum - at[i] - bt[i];

            tat = tat + sum - at[i];

            count = 0;

        }

        if (i == NOP - 1)

        {

            i = 0;

        }

        else if (at[i + 1] <= sum)

        {

            i++;

        }

        else

        {

            i = 0;

        }

    }

    avg\_wt = wt \* 1.0 / NOP;

    avg\_tat = tat \* 1.0 / NOP;

    printf("\n Average Turn Around Time: \t%f", avg\_wt);

    printf("\n Average Waiting Time: \t%f", avg\_tat);

    getchar();

}

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

*// PS C:\Users\Krishnendu Das\OneDrive\Desktop\Shell and OS> gcc rr.c*

*// PS C:\Users\Krishnendu Das\OneDrive\Desktop\Shell and OS> ./a.exe*

*//  Total number of process in the system: 5*

*//  Enter the Arrival and Burst time of the Process[1]*

*//  Arrival time is:       4*

*// Burst time is:  6*

*//  Enter the Arrival and Burst time of the Process[2]*

*//  Arrival time is:       3*

*// Burst time is:  5*

*//  Enter the Arrival and Burst time of the Process[3]*

*//  Arrival time is:       4*

*// Burst time is:  7*

*//  Enter the Arrival and Burst time of the Process[4]*

*//  Arrival time is:       2*

*// Burst time is:  5*

*//  Enter the Arrival and Burst time of the Process[5]*

*//  Arrival time is:       6*

*// Burst time is:  8*

*// Enter the Time Quantum for the process:         4*

*//  Process No              Burst Time              TAT             Waiting Time*

*// Process No[1]            6                       18                      12*

*// Process No[2]            5                       20                      15*

*// Process No[3]            7                       22                      15*

*// Process No[4]            5                       25                      20*

*// Process No[5]            8                       25                      17*

*//  Average Turn Around Time:      15.800000*

*//  Average Waiting Time:  22.000000*

*// PS C:\Users\Krishnendu Das\OneDrive\Desktop\Shell and OS>*

# Program-13

*# Write a shell program to print the roots of a quadratic equation.*

*#Submitted By Roll No :- 302*

*#Source Code*

*#!bin/bash*

echo Enter the coefficient of x^2:

read a

echo Enter the coefficient of x:

read b

echo Enter the constant term:

read c

f=`echo "-($b)" |bc`

p=`expr 2 \\* $a`

if [ $a -ne 0 ]

then

    d=`echo \( \( $b \\* $b \) - \( 4 \\* $a \\* $c \) \) | bc`

    if [ $d -lt 0 ]

    then

        x=`echo "-($d)" | bc`

        s=`echo "scale=2; sqrt ( $x )" | bc`

        echo The first root is:

        echo "($f + $s i) / $p"

        echo The second root is:

        echo "($f - $s i) / $p"

    elif [ $d -eq 0 ]

    then

        res=`expr $f / $p`

        echo The root is: $res

    else

        s=`echo "scale=2; sqrt( $d )" | bc`

        res1=`echo "scale=2; ( $f + $s) / ( $p )"|bc`

        res2=`echo "scale=2; ( $f - $s) / ( $p )"|bc`

        echo The first root is: $res1

        echo The second root is: $res2

    fi

else

    echo Coefficient of x^2 can not be 0.

fi

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

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$ bash quadratic.sh*

*# Enter the coefficient of x^2:*

*# 4*

*# Enter the coefficient of x:*

*# 6*

*# Enter the constant term:*

*# 2*

*# The first root is: -.50*

*# The second root is: -1.00*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$*

# Program-14

*# Write a shell program to implement a menu driven calculator using switch case.*

*#Submitted By Roll No 302*

*#Source Code*

sum=0

i="y"

echo "Enter first number :"

read n1

echo "Enter second number :"

read n2

while [ $i = "y" ]

do

echo "1.Addition"

echo "2.Subtraction"

echo "3.Multiplication"

echo "4.Division"

echo "Enter your choice"

read ch

case $ch in

1)sum=`expr $n1 + $n2`

echo "Sum ="$sum;;

2)sub=`expr $n1 - $n2`

echo "Sub = "$sub;;

3)mul=`expr $n1 \\* $n2`

echo "Mul = "$mul;;

4)div=`echo $n1 / $n2 | bc -l`

echo "Div = "$div;;

\*)echo "Invalid choice";;

esac

echo "Do u want to continue ?"

read i

if [ $i != "y" ]

then

exit

fi

done

#########################OUTPUT#################

# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$ bash calculator.sh

# Enter first number :

# 6

# Enter second number :

# 8

# 1.Addition

# 2.Subtraction

# 3.Multiplication

# 4.Division

# Enter your choice

# 1

# Sum =14

# Do u want to continue ?

# y

# 1.Addition

# 2.Subtraction

# 3.Multiplication

# 4.Division

# Enter your choice

# 2

# Sub = -2

# Do u want to continue ?

# y

# 1.Addition

# 2.Subtraction

# 3.Multiplication

# 4.Division

# Enter your choice

# 3

# Mul = 48

# Do u want to continue ?

# y

# 1.Addition

# 2.Subtraction

# 3.Multiplication

# 4.Division

# Enter your choice

# 4

# Div = .75000000000000000000

# Do u want to continue ?

# n

# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$

# Program-15

*#Write a shell program to check whether a given date is valid or not.*

*#Year should be between 0 to 2022, month should be between 1 to 12, date should be between 1 to 31 with leap year check.*

*#Use switch case for checking the range.*

*#Submitted by Roll No : 302*

*#Source Code*

*#!/bin/bash*

dd=0

mm=0

yy=0

days=0

read -p "Enter day (dd): " dd

read -p "Enter month (mm): " mm

read -p "Enter year (yyyy): " yy

if [ $mm -le 0 -o $mm -gt 12 ]

then

    echo "$mm is invalid month."

    exit 1

fi

case $mm in

    1) days=31;;

    2) days=28 ;;

    3) days=31 ;;

    4) days=30 ;;

    5) days=31 ;;

    6) days=30 ;;

    7) days=31 ;;

    8) days=31 ;;

    9) days=30 ;;

    10) days=31 ;;

    11) days=30 ;;

    12) days=31 ;;

    \*) days=-1;;

esac

if [ $mm -eq 2 ];

then

    if [ $((yy % 4)) -ne 0 ]

        then :

    elif [ $((yy % 400)) -eq 0 ]

        then

            days=29

    elif [ $((yy % 100)) -eq 0 ]

        then :

    else

       days=29

    fi

fi

if [ $dd -le 0 -o $dd -gt $days ];

then

    echo "$dd day is invalid"

    exit 3

fi

if [ $yy -le 0 -o $yy -gt 2022 ];

then

    echo "$yy year is invalid"

    exit 4

fi

echo "$dd/$mm/$yy is a vaild date"

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

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$ bash date.sh*

*# Enter day (dd): 29*

*# Enter month (mm): 2*

*# Enter year (yyyy): 2004*

*# 29/2/2004 is a vaild date*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$ bash date.sh*

*# Enter day (dd): 29*

*# Enter month (mm): 2*

*# Enter year (yyyy): 2000*

*# 29/2/2000 is a vaild date*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$ bash date.sh*

*# Enter day (dd): 29*

*# Enter month (mm): 2*

*# Enter year (yyyy): 2013*

*# 29 day is invalid*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$*

# Program-16

*# Write a shell program to find all the lines in a file containing a word. After dropping those line, store the remaining text in an output file.*

*#Submitted By Roll No:-302*

*#Source Code*

*#!/bin/bash*

echo "enter file name"

read file

echo "enter word"

read word

echo "file before removing" $word:

cat $file

grep -v -i $word $file > test.txt

mv test.txt $file

echo "file after removing" $word:

cat $file

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

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$ bash grep.sh*

*# enter file name*

*# AVG.sh*

*# enter word*

*# echo*

*# file before removing echo:*

*# read N*

*# i=1*

*# sum=0*

*# while [ $i -le $N ]*

*# do*

*#   read num*

*#   sum=$((sum + num))*

*#   i=$((i + 1))*

*# done*

*# file after removing echo:*

*# read N*

*# i=1*

*# sum=0*

*# while [ $i -le $N ]*

*# do*

*#   read num*

*#   sum=$((sum + num))*

*#   i=$((i + 1))*

*# done*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$*

# Program-17

*#Write a shell program to find the maximum number in an array.*

*#Submitted by Roll No : 302*

*#Source Code*

*#!/bin/bash*

clear

read -p "Enter size of array: " n

echo "Enter numbers: "

i=0

max=0

while [ $i -lt $n ]

do

    read arr[$i]

    if [ $i -eq 0 ]

    then

        max=${arr[$i]}

    else

        if [ ${arr[$i]} -gt $max ]

        then

            max=${arr[$i]}

        fi

    fi

    ((i++))

done

echo "The maximum number in the array is" $max

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

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$* bash maxarr.sh

*#Enter size of array: 5*

*#Enter numbers:*

*#3*

*#6*

*#9*

*#8*

*#7*

*#The maximum number in the array is 9*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$*

# Program-18

*# Write a shell program to perform linear search.*

*#Submitted By Roll No 302*

*#Source Code*

clear

read -p "Enter the number of elements:" n

echo "Enter the elements:"

i=0

while [ $i -lt $n ]

do

read arr[$i]

((i++))

done

read -p "Enter the element to search:" s

temp=${arr[$i]}

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

do

while [ ${arr[$i]} -eq $s ]

do

echo "Element Found!!"

break

done

done

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

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$*bash linearsearch.sh

*# Enter the number of elements:6*

*# Enter the elements:*

*# 5*

*# 8*

*# 3*

*# 1*

*# 2*

*# 9*

*# Enter the element to search:5*

*# Element Found!!*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$*

# Program-19

*# Write a shell program to reverse an array.*

*#Submitted By Roll No:-302*

*#Source Code*

*#!/bin/bash*

declare -a array

read -p "Enter the length of array list: " n

echo "Enter $n elements: "

for (( i=0;i<$n;i++))

do

    read elements

    array[$i]="$elements"

done

min=0

max=$(( ${#array[@]} -1 ))

while [[ min -lt max ]]

do

    x="${array[$min]}"

    array[$min]="${array[$max]}"

    array[$max]="$x"

    (( min++, max-- ))

done

echo "Reverce Array is :${array[@]}"

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

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$ bash revarray.sh*

*# Enter the length of array list: 5*

*# Enter 5 elements:*

*# 4*

*# 1*

*# 3*

*# 7*

*# 8*

*# Reverce Array is :8 7 3 1 4*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$*

# Program-20

*# Write a shell program to implement stack using array.*

*#Submitted By :- 302*

*#Source Code*

*#!bin/bash*

declare -a stack

top=-1

max=$1

echo -e "Stack Size : $max"

((max=max-1))

push(){

    if [ $top -eq $max ]

    then

        echo "Stack Overflow"

        exit 1

    elif [ $top -lt $max ]

    then

        ((top=top+1))

        stack[$top]=$1

    else

        echo "Stack Overflow"

        exit 1

    fi

}

pop(){

    if [ $top -gt -1 ]

    then

        echo "${stack[$top]} Popped"

        unset stack[$top]

        ((top=top-1))

    else

        echo "Stack Underflow"

        exit 1

    fi

}

while [ 1 -eq 1 ]

do

    echo -e "\n1.Push\n2.Pop\n3.Exit"

    read -p "Enter Your Choice : " choice

    case $choice in

        1)

            read -p "Enter the element : " element

            push $element

            echo "Stack : ${stack[\*]}"

        ;;

        2)

            pop

            echo "Stack : ${stack[\*]}"

        ;;

        3)

            echo "Exiting"

            exit 1

        ;;

        \*) echo "Invalid input..!"

        ;;

    esac

done

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

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$ bash stack.sh 5*

*# Stack Size : 5*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 1*

*# Enter the element : 2*

*# Stack : 2*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 1*

*# Enter the element : 3*

*# Stack : 2 3*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 1*

*# Enter the element : 4*

*# Stack : 2 3 4*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 1*

*# Enter the element : 5*

*# Stack : 2 3 4 5*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 1*

*# Enter the element : 6*

*# Stack : 2 3 4 5 6*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 2*

*# 6 Popped*

*# Stack : 2 3 4 5*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 2*

*# 5 Popped*

*# Stack : 2 3 4*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 2*

*# 4 Popped*

*# Stack : 2 3*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 2*

*# 3 Popped*

*# Stack : 2*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 2*

*# 2 Popped*

*# Stack :*

*# 1.Push*

*# 2.Pop*

*# 3.Exit*

*# Enter Your Choice : 2*

*# Stack Underflow*

*# krishnendu@krishnendu-OptiPlex-3046:~/Desktop/Shell$*