1. PROGRAM TO FIND GREATEST OF TWO NUMBERS

Aim:

To write a program to find the greatest of two numbers

Algorithm:

1. Enter the two numbers which are to be compared.
2. Check for the greatest number among the two.
3. If A is greater, then print the result as “A is the greatest number”.
4. If B is greater, then print the result as “B is the greatest number”.

Program:

```
echo “enter 2 numbres”

read a b

if[$a-$b]

then

echo “$a is greater”

else

echo “$b is greater”

fi
```

Result:

Thus the program to find the greater f two numbers was successfully executed.

2. Find the cube

Aim:

To write a program to find the cube

Algorithm:

1. Enter the value of a number
2. Perform the product of the given number thrice
3. Display the result.

Program:

```
echo “enter a number”

read a

let b=$a\*$a\*$a

echo “the cube is $b”
```

Output:

Finding the queue:

Enter a number

3

The cube is 27

Result:

Thus the above program finding the cube has been executed successfully.

3.Sum of N numbers:

Aim:

To write a program for sum of N numbers

Algorithm:

1. Enter the limit as n.
2. Initialize I as 1 and sum as 0.
3. Compute the value of sum and I until I is less than or equal to n.
4. Sum is computed by adding the value of sum and i.
5. I is computed by adding one to i. and display the result of sum.

Program:

```
echo “enter limit”

read n

i=1

sum=0

while[$i –le $n]

do

let sum=$sum+$i

let i=$i+1

done

echo”the sum of $n numbers is $sum”
```

Output:

```
Sum of N numbers

Enter limit 8

The sum of 8 numbers is 36
```

Result:

Thus the above program sum of N numbers has been executed successfully.

4. Swapping of two numbers

Aim:

To write a program to swapping of two numbers

Algorithm:

1. Enter the 2 numbers such as a and b to be swapped.
2. Assign the temp value as one.
3. Assign the temp variable as a.
4. Assign the value of b to a.
5. Assign the temp value to b.
6. Display the value of a and b.

Program:

```
echo “enter two numbers”

read a b

t=$a

a=$b

b=$t

b=$t

echo “a=$a b=$b”
```

Output:

Swapping of two numbers:

Enter two numbers 5 9 a=9 b=5

Result:

Thus the above program swapping of two numbers has been executed successfully.

5. Checking the number is positive or negative

Aim:

To write a program to find the given number is positive or negative

Algorithm:

1. Enter the number
2. If the number is greater than zero, display that the number is positive.
3. If the number is less than zero, display that the number is negative.
4. If the number is neither greater than zero nor less than zero, display that the number is zero.

Program:

```
echo" enter the number"

read a

if[$a –ge 0]

then

echo "$a is positive"

else

echo "$a is negative"

fi
```

Output:

```
Checking the number is positive or negative

7 is positive

Enter a number

-8

-8 is negative
```

Result:

Thus the above program has been executed successfully.

Ex No: 6 Basic Calculator Using Switch Case

Aim:

To develop a Basic Math calculator Using case Statement.

Algorithm:

1) Create a new file.

2) Read the operands.

3) Select any one operation from the list.

4) Perform the operation. 5) Print the result.

Program:

```
# Implementation of Calculator application

#!bin/bash

j=1

while [ $j -eq 1 ]

do

echo &quot;Enter the First Operand;&quot;

read f1

echo &quot;Enter the second operand:&quot;

read f2

echo &quot;1-&gt; Addition&quot;

echo &quot;2-&gt; Subtraction&quot;

echo &quot;3-&gt; Multiplication&quot;

echo &quot;4-&gt; Division&quot;

echo &quot;Enter your choice&quot;

read n

case &quot;$n&quot; in
```

```
1)
echo &quot;Addition&quot;
f3=$((f1+f2))
echo &quot;The result is:$f3&quot;
;;
2)
echo &quot;Subtraction&quot;
let &quot;f4=$f1 - $f2&quot;
echo &quot;The result is:$f4&quot;
;;
3)
echo &quot;Multiplication&quot;
let &quot;f5=$f1 * $f2&quot;
echo &quot;The result is:$f5&quot;
;;
4)

echo &quot;Division&quot;
let &quot;f6=$f1 / $f2&quot;
echo &quot;The result is:$f6&quot;
;;
esac
echo &quot;Do you want to continue(press:1 otherwise press any key to
quit)&quot;
read j done
```

Result:

Thus the above program to develop a calculator application was executed successfully.

7. First come First Serve

Aim:

To write a c program for FCFS

Algorithm:

1. Get the number of processes and burst time.
2. The process is executed in the order given by the user.
3. Calculate the waiting time and turnaround time.
4. Display the gantt chart, avg waiting time and turnaround time.

Program:

```
#include&lt;stdio.h&gt;

int main()

{

int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;

printf(&quot;Enter total number of processes(maximum 20):&quot;);

scanf(&quot;%d&quot;,&amp;n);

printf(&quot;nEnter Process Burst Timen&quot;);

for(i=0;i&lt;n;i++)

{

printf(&quot;\n P[%d]:&quot;,i+1);

scanf(&quot;%d&quot;,&amp;bt[i]);

}

wt[0]=0;

for(i=1;i&lt;n;i++)

{

wt[i]=0;

for(j=0;j&lt;i;j++)

wt[i]+=bt[j];
```

```
}
printf(&quot;ProcessttBurst TimetWaiting TimetTurnaround Time&quot;);
for(i=0;i&lt;n;i++)
{
tat[i]=bt[i]+wt[i];
avwt+=wt[i];
avtat+=tat[i];
printf(&quot;\n P[%d]tt%dtt%dtt%d&quot;,i+1,bt[i],wt[i],tat[i]);
}
avwt/=i;
avtat/=i;
printf(&quot;\n Average Waiting Time:%d&quot;,avwt);
printf(&quot;\n Average Turnaround Time:%d&quot;,avtat);
return 0;
}
```

Output:

```
Enter total number of processes (maximum 20):3
Enter Process Burst Time
P[1]:1
P[2]:2
P[3]:3
ProcessttBurst TimetWaiting TimetTurnaround Time
P[1]tt1tt0tt1
P[2]tt2tt1tt3
P[3]tt3tt3tt6
Average Waiting Time:1
```

Average Turnaround Time:3

Result:

Thus the above program has been executed successfully.

8. Shortest Job First

Aim:

To write a c program for SJF

Algorithm:

1. Get the number of processes and burst time.
2. Sort the process based on the burst time in ascending order.
3. Calculate the waiting time and turnaround time.
4. Display the gantt chart, avg waiting time and turnaround time.

Program:

```
#include&lt;stdio.h&gt;

 int main()

{

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

   float avg_wt,avg_tat;

   printf(&quot;\nEnter number of process:&quot;);

   scanf(&quot;%d&quot;,&amp;n);


   printf(&quot;\nEnter Burst Time:n&quot;);

   for(i=0;i&lt;n;i++)

   {

      printf(&quot;p%d:&quot;,i+1);

      scanf(&quot;%d&quot;,&amp;bt[i]);

      p[i]=i+1;

   }


  //sorting of burst times

   for(i=0;i&lt;n;i++)

   {
```

```
    pos=i;
    for(j=i+1;j&amp;lt;n;j++)
    {
        if(bt[j]&amp;lt;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&amp;lt;n;i++)
{
    wt[i]=0;
    for(j=0;j&amp;lt;i;j++)
        wt[i]+=bt[j];

    total+=wt[i];
}

avg_wt=(float)total/n;
total=0;
```

```
   printf(&quot;\nProcesst    Burst Time    tWaiting TimetTurnaround
Time&quot;);
   for(i=0;i&lt;n;i++)
   {
      tat[i]=bt[i]+wt[i];
      total+=tat[i];
      printf(&quot;np%dtt  %dtt
%dttt%d&quot;,p[i],bt[i],wt[i],tat[i]);
   }

   avg_tat=(float)total/n;
   printf(&quot;\n Average Waiting Time=%f&quot;,avg_wt);
   printf(&quot;\ nAverage Turnaround
}
```

Output:

SJF:

Enter the number of process 3

Enter the burst time 2 1 3

Gantt chart

P1|p2|p3|

0 1 3 6

Average waiting time is 1.33

Average turnaround time is 3.33

Result:

Thus the above program has been executed successfully.

9.Round Robin(pre-emptive)

Aim:

To write a c program for round robin algorithm

Algorithm:

1. Get the number of processes and burst time.
2. Sort the process based on the burst time in ascending order.
3. Calculate the waiting time and turnaround time.
4. Display the gantt chart, avg waiting time and turnaround time.

Program

```
#include<stdio.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(&quot; Total number of process in the system: &quot;);

scanf(&quot;%d&quot;, &amp;NOP);

y = NOP;

for(i=0; i&lt;NOP; i++)

{

printf(&quot;\n Enter the Arrival and Burst time of the

Process[%d]\n&quot;, i+1);

printf(&quot; Enter Arrival time: \t&quot;);

scanf(&quot;%d&quot;, &amp;at[i]);

printf(&quot; \nEnter Burst time: \t&quot;);

scanf(&quot;%d&quot;, &amp;bt[i]);

temp[i] = bt[i];

}
```

```
printf(&quot;Enter the Time Quantum for the process: \t&quot;);
scanf(&quot;%d&quot;, &amp;quant);
printf(&quot;\n Process No \t\t Burst Time \t\t TAT \t\t Waiting
Time &quot;);
for(sum=0, i = 0; y!=0; )
{
if(temp[i] &lt;= quant &amp;&amp; temp[i] &gt; 0)
{
sum = sum + temp[i];
temp[i] = 0;
count=1;
}
else if(temp[i] &gt; 0)
{
temp[i] = temp[i] - quant;
sum = sum + quant;
}
if(temp[i]==0 &amp;&amp; count==1)
{
y--;

printf(&quot;\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t
%d&quot;, 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]&lt;=sum)
{
i++;
}
else
{
i=0;
}
}
}
```

Output:

Total number of process in the system: 2

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

Enter Arrival time: 22

Enter Burst time: 3

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

Enter Arrival time: 2

Enter Burst time: 2

Enter the Time Quantum for the process: 3

Process No Burst Time TAT Waiting

Time

Process No[1] 3 -19 -22

Process No[2] 2 3 1

Result:

Thus the above program has been executed successfully.

10. Priority

Aim:

To write a program for priority scheduling

Algorithm:

1. Get the number of processes and burst time.
2. Sort the process based on the burst time in ascending order.
3. Calculate the waiting time and turnaround time.
4. Display the gantt chart, avg waiting time and turnaround time.

Program

```
#include<stdio.h>

//Function to swap two variables

void swap(int *a,int *b)

{

int temp=*a;

*a=*b;

*b=temp;

}

int main()

{

int n;

printf(&quot;Enter Number of Processes: &quot;);

scanf(&quot;%d&quot;,&amp;n);

// b is array for burst time, p for priority and index for

process id

int b[n],p[n],index[n];

for(int i=0;i&lt;n;i++)

{

printf(&quot;Enter Burst Time and Priority Value for Process %d:

&quot;,i+1);
```

```
scanf(&quot;%d %d&quot;,&amp;b[i],&amp;p[i]);
index[i]=i+1;
}
for(int i=0;i&lt;n;i++)
{
int a=p[i],m=i;
//Finding out highest priority element and placing it at
its desired position
for(int j=i;j&lt;n;j++)
{
if(p[j] &gt; a)
{
a=p[j];

m=j;
}
}

//Swapping processes
swap(&amp;p[i], &amp;p[m]);
swap(&amp;b[i], &amp;b[m]);
swap(&amp;index[i],&amp;index[m]);
}

// T stores the starting time of process
int t=0;

//Printing scheduled process
```

```
printf(&amp;quot;Order of process Execution is\n&amp;quot;);
for(int i=0;i&amp;lt;n;i++)
{
printf(&amp;quot;P%d is executed from %d to
%d\n&amp;quot;,index[i],t,t+b[i]);
t+=b[i];
}
printf(&amp;quot;\n&amp;quot;);
printf(&amp;quot;Process Id Burst Time Wait Time TurnAround
Time\n&amp;quot;);
int wait_time=0;

for(int i=0;i&amp;lt;n;i++)
{
printf(&amp;quot;P%d %d %d
%d\n&amp;quot;,index[i],b[i],wait_time,wait_time + b[i]);
wait_time += b[i];
}
return 0;
}
Enter Number of Processes: 4
Enter Burst Time and Priority Value for Process 1: 1
2
Enter Burst Time and Priority Value for Process 2: 2
2
Enter Burst Time and Priority Value for Process 3: 2
3
Enter Burst Time and Priority Value for Process 4: 3
```

3

Order of process Execution is

P3 is executed from 0 to 2

P4 is executed from 2 to 5

P1 is executed from 5 to 6

P2 is executed from 6 to 8

Process Id Burst Time Wait Time TurnAround Time

P3 2 0 2

P4 3 2 5

P1 1 5 6

P2 2 6 8

Result:

Thus the above program has been executed successfully.