

# **OPERATING SYSTEMS ASSIGNMENT – 01**

**Name:** Kamisha Salim

**S.ID:** 11070

## QUESTION - 1

Part - a: Define priority scheduling algorithm and also its advantages and disadvantages.

Priority Scheduling Algorithm:- It is a method of scheduling processes that is based on priority. In this algorithm, the scheduler selects the tasks to work as per the priority. The processes with higher priority are carried out first, whereas jobs with equal priorities are carried out on a round robin or first come first serve (FCFS) basis.

### Advantages:-

- 1) This provides a good mechanism where the relative importance of each process is precisely defined.
- 2) Processes are executed on the basis of priority, so high priority does not need to wait for a long time.
- 3) It is easy to use and simple to understand.

### Disadvantages:-

- 1) If high priority processes uses up a lot of <sup>CPU</sup> time, lower priority processes may starve and be postponed indefinitely.
- 2) In case of having processes of the same priority, then we have to make use of another scheduling algorithm.
- 3) If the system crashes, processes with lower priority that were not finished yet will get lost.

**Part – b: Write a program of the given algorithm. Apply priority scheduling algorithm using C/C++.**

**CODE:**

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 struct Process {
4     int pid;
5     int bt;
6     int priority;
7 };
8 bool comp(Process a, Process b) {
9     return (a.priority > b.priority);
10 }
11 void waitingtime(Process pro[], int n, int wt[]) {
12     wt[0] = 0;
13     for (int i = 1; i < n; i++)
14         wt[i] = pro[i-1].bt + wt[i-1];
15 }
16 void turnaroundtime(Process pro[], int n, int wt[], int tat[]) {
17     for (int i = 0; i < n; i++)
18         tat[i] = pro[i].bt + wt[i];
19 }
20 void avgtime(Process pro[], int n) {
21     int wt[n], tat[n], total_wt = 0, total_tat = 0;
22     waitingtime(pro, n, wt);
23     turnaroundtime(pro, n, wt, tat);
24     cout << "\nProcesses "<< "Burst time " << " Waiting time " << " Turn around time\n";
25     for (int i = 0; i < n; i++) {
26         total_wt = total_wt + wt[i];
27         total_tat = total_tat + tat[i];
28         cout << " " << pro[i].pid << "\t\t" << pro[i].bt << "\t " << wt[i] << "\t\t" << tat[i] << endl;
29     }
30     cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
31     cout << "\nAverage turn around time = " << (float)total_tat / (float)n;
32 }
33 void scheduling(Process pro[], int n) {
34     sort(pro, pro + n, comp);
35     cout<< "Order in which processes gets executed \n";
36     for (int i = 0; i < n; i++)
37         cout << pro[i].pid << " ";
38     avgtime(pro, n);
39 }

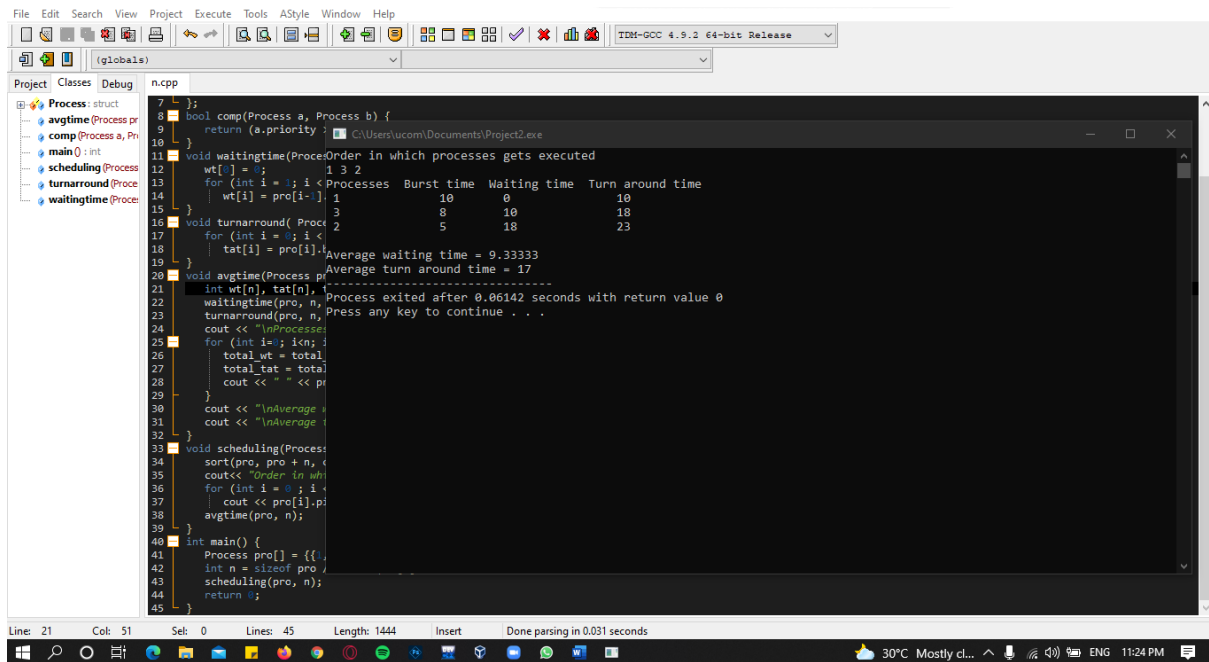
```

```

7 };
8 bool comp(Process a, Process b) {
9     return (a.priority > b.priority);
10 }
11 void waitingtime(Process pro[], int n, int wt[]) {
12     wt[0] = 0;
13     for (int i = 1; i < n; i++)
14         wt[i] = pro[i-1].bt + wt[i-1];
15 }
16 void turnaroundtime(Process pro[], int n, int wt[], int tat[]) {
17     for (int i = 0; i < n; i++)
18         tat[i] = pro[i].bt + wt[i];
19 }
20 void avgtime(Process pro[], int n) {
21     int wt[n], tat[n], total_wt = 0, total_tat = 0;
22     waitingtime(pro, n, wt);
23     turnaroundtime(pro, n, wt, tat);
24     cout << "\nProcesses "<< "Burst time " << " Waiting time " << " Turn around time\n";
25     for (int i = 0; i < n; i++) {
26         total_wt = total_wt + wt[i];
27         total_tat = total_tat + tat[i];
28         cout << " " << pro[i].pid << "\t\t" << pro[i].bt << "\t " << wt[i] << "\t\t" << tat[i] << endl;
29     }
30     cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
31     cout << "\nAverage turn around time = " << (float)total_tat / (float)n;
32 }
33 void scheduling(Process pro[], int n) {
34     sort(pro, pro + n, comp);
35     cout<< "Order in which processes gets executed \n";
36     for (int i = 0; i < n; i++)
37         cout << pro[i].pid << " ";
38     avgtime(pro, n);
39 }
40 int main() {
41     Process pro[] = {{1, 10, 2}, {2, 5, 0}, {3, 8, 1}};
42     int n = sizeof pro / sizeof pro[0];
43     scheduling(pro, n);
44     return 0;
45 }

```

## OUTPUT:



```
7 }
8 bool comp(Process a, Process b) {
9     return (a.priority > b.priority);
10 }
11 void waitingtime(ProceOrder in which processes gets executed
12     wt[] = {0};
13     for (int i = 1; i < Processes; i++)
14         wt[i] = pro[i-1].burst + wt[i-1];
15 }
16 void turnarround(Proce
17     for (int i = 0; i < n; i++)
18         tat[i] = pro[i].wt + pro[i].burst;
19 }
20 void avgtime(Process p
21     int wt[n], tat[n],
22     waitingtime(pro, n,
23     turnarround(pro, n,
24     cout << "Processes
25     for (int i = 1; i < n; i++)
26         total_wt = total_wt + wt[i];
27         total_tat = total_tat + tat[i];
28     }
29     cout << "Average waiting time = 9.33333
30     cout << "Average turn around time = 17
31 }
32 }
33 void scheduling(Process
34     sort(pro, pro + n, comp);
35     cout << "Order in which processes gets executed:
36     for (int i = 0; i < n; i++)
37         cout << pro[i].id << " ";
38     avgtime(pro, n);
39 }
40 int main() {
41     Process pro[] = {{1, 10, 3}, {2, 5, 2}, {3, 8, 1}};
42     int n = sizeof pro / sizeof Process;
43     scheduling(pro, n);
44     return 0;
45 }
```

Process: struct  
avgtime(Process p  
comp(Process a, p  
main() : int  
scheduling(Process  
turnarround(Proce  
waitingtime(Proce

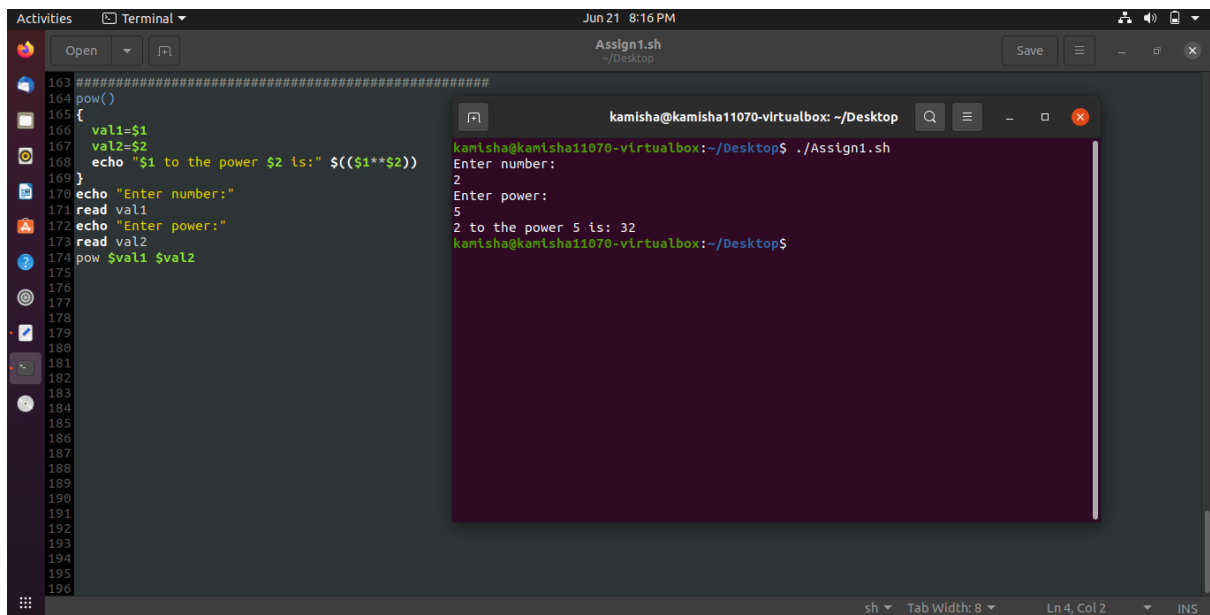
Order in which processes gets executed  
1 3 2  
Processes Burst time Waiting time Turn around time  
1 10 0 10  
2 5 18 23  
3 8 10 18  
Average waiting time = 9.33333  
Average turn around time = 17  
Process exited after 0.06142 seconds with return value 0  
Press any key to continue . . .

Line: 21 Col: 51 Sel: 0 Lines: 45 Length: 1444 Insert Done parsing in 0.031 seconds

30°C Mostly cl... ENG 11:24 PM

## QUESTION – 2

### Part – a: Print a power b



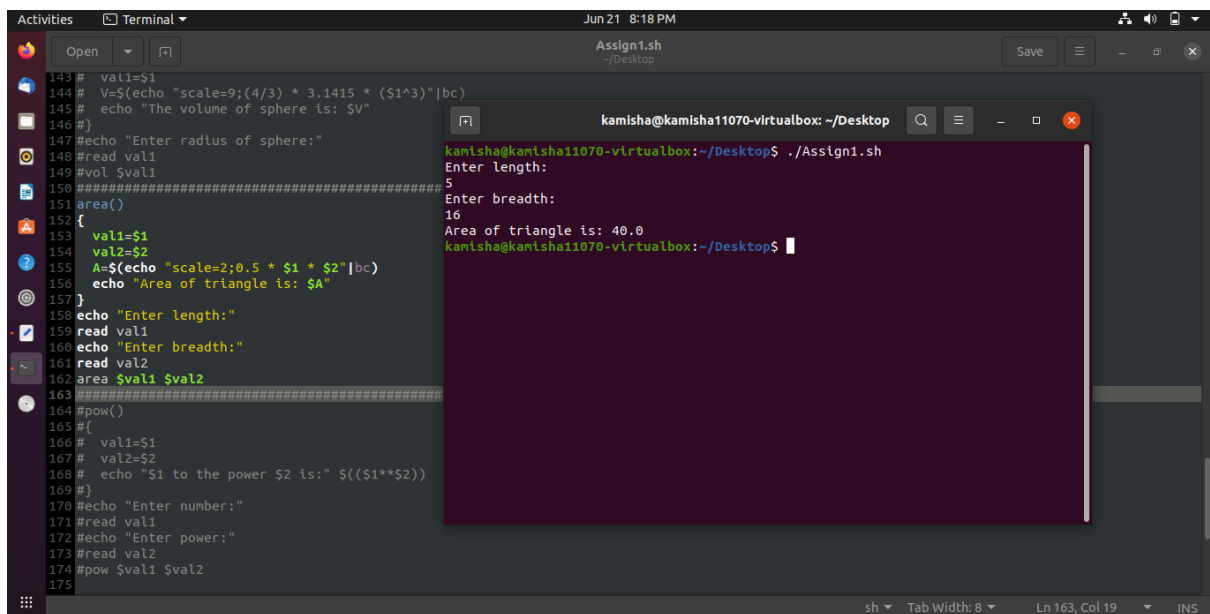
The screenshot shows a terminal window with a script named `Assign1.sh` and its execution. The script prompts the user for a number and a power, then calculates the power using the `pow` command.

```
163 #####
164 pow()
165 {
166     val1=$1
167     val2=$2
168     echo "$1 to the power $2 is:" `${1**$2}`
169 }
170 #echo "Enter number:"
171 read val1
172 #echo "Enter power:"
173 read val2
174 pow $val1 $val2
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
```

Execution output:

```
kamisha@kamisha11070-virtualbox: ~/Desktop
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter number:
2
Enter power:
5
2 to the power 5 is: 32
kamisha@kamisha11070-virtualbox:~/Desktop$
```

### Part – b: Calculate area of triangle



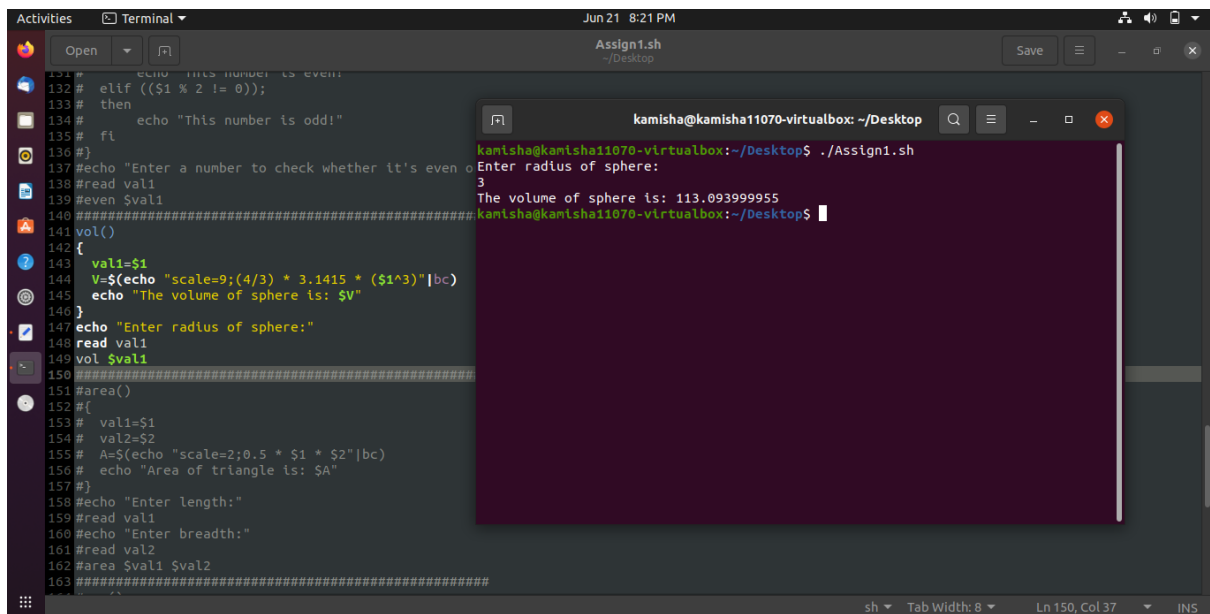
The screenshot shows a terminal window with a script named `Assign1.sh` and its execution. The script prompts the user for the length and breadth of a triangle, then calculates the area using the formula  $A = \frac{1}{2} \times \text{length} \times \text{breadth}$ .

```
143 # val1=$1
144 # V=$(echo "scale=9;(4/3) * 3.1415 * (${1^3})"|bc)
145 # echo "The volume of sphere is: $V"
146 #}
147 #echo "Enter radius of sphere:"
148 #read val1
149 #vol $val1
150 #####
151 area()
152 {
153     val1=$1
154     val2=$2
155     A=$(echo "scale=2;0.5 * $1 * $2"|bc)
156     echo "Area of triangle is: $A"
157 }
158 #echo "Enter length:"
159 read val1
160 #echo "Enter breadth:"
161 read val2
162 area $val1 $val2
163 #####
164 #pow()
165 # {
166 #     val1=$1
167 #     val2=$2
168 #     echo "$1 to the power $2 is:" `${1**$2}`
169 # }
170 #echo "Enter number:"
171 #read val1
172 #echo "Enter power:"
173 #read val2
174 #pow $val1 $val2
175
```

Execution output:

```
kamisha@kamisha11070-virtualbox: ~/Desktop
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter length:
5
Enter breadth:
16
Area of triangle is: 40.0
kamisha@kamisha11070-virtualbox:~/Desktop$
```

## Part – c: Calculate volume of sphere

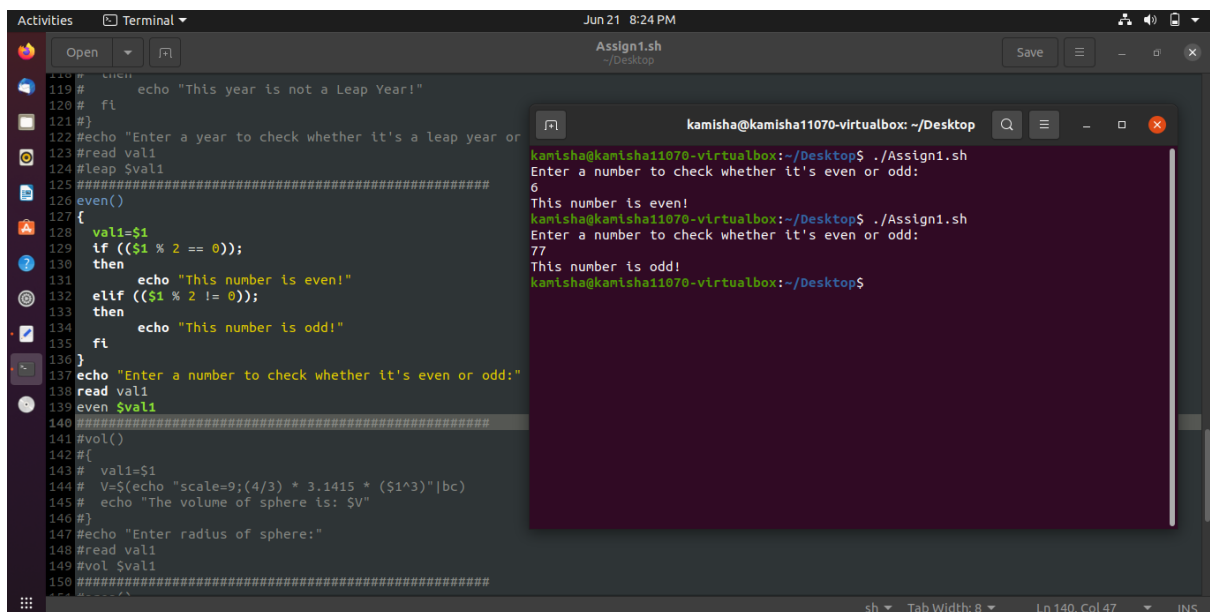


```
131# elif (($1 % 2 != 0));
132# then
133#   echo "This number is odd!"
134# fi
135#}
136#}
137# echo "Enter a number to check whether it's even or odd:"
138# read val1
139# even $val1
140#####
141# vol()
142#{
143#   val1=$1
144#   V=$(echo "scale=9;(4/3) * 3.1415 * ($1^3)"|bc)
145#   echo "The volume of sphere is: $V"
146#}
147# echo "Enter radius of sphere:"
148# read val1
149# vol $val1
150#####
151# area()
152#{
153#   val1=$1
154#   val2=$2
155#   A=$(echo "scale=2;0.5 * $1 * $2"|bc)
156#   echo "Area of triangle is: $A"
157#}
158# echo "Enter length:"
159# read val1
160# echo "Enter breadth:"
161# read val2
162# area $val1 $val2
163#####
```

Terminal Output:

```
kamisha@kamisha11070-virtualbox: ~/Desktop
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter radius of sphere:
3
The volume of sphere is: 113.093999955
kamisha@kamisha11070-virtualbox:~/Desktop$
```

## Part – d: Check if a number is odd or even

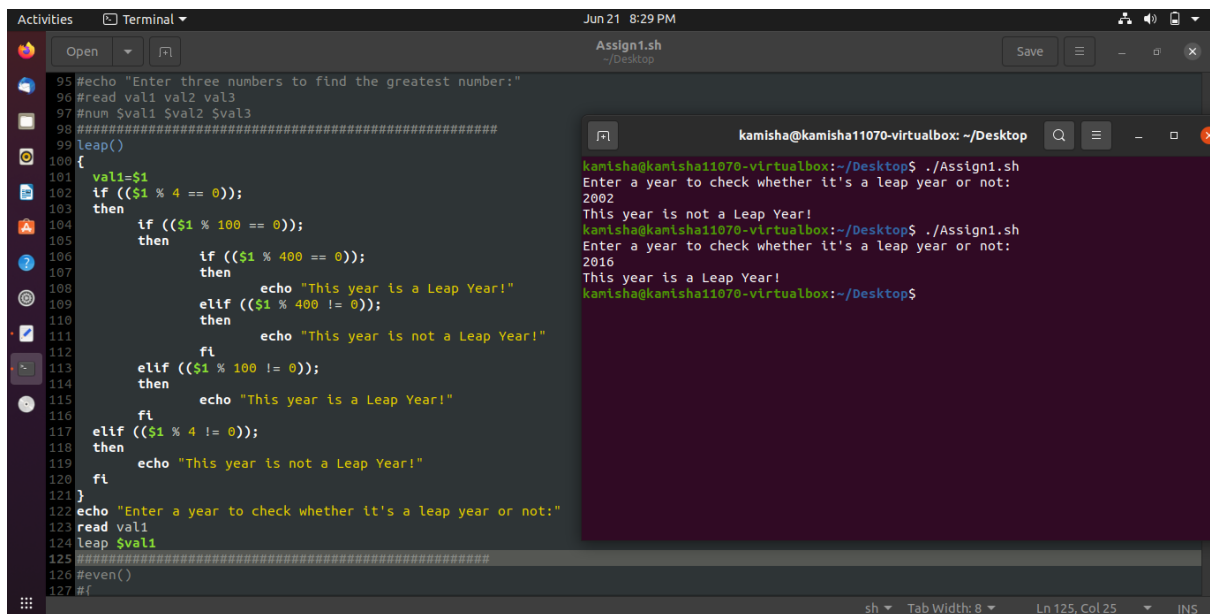


```
119#   echo "This year is not a Leap Year!"
120# fi
121#}
122# echo "Enter a year to check whether it's a leap year or not:"
123# read val1
124# leap $val1
125#####
126# even()
127#{
128#   val1=$1
129#   if (($1 % 2 == 0));
130#   then
131#       echo "This number is even!"
132#   elif (($1 % 2 != 0));
133#   then
134#       echo "This number is odd!"
135#   fi
136#}
137# echo "Enter a number to check whether it's even or odd:"
138# read val1
139# even $val1
140#####
141# vol()
142#{
143#   val1=$1
144#   V=$(echo "scale=9;(4/3) * 3.1415 * ($1^3)"|bc)
145#   echo "The volume of sphere is: $V"
146#}
147# echo "Enter radius of sphere:"
148# read val1
149# vol $val1
150#####
```

Terminal Output:

```
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter a number to check whether it's even or odd:
6
This number is even!
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter a number to check whether it's even or odd:
77
This number is odd!
kamisha@kamisha11070-virtualbox:~/Desktop$
```

## Part – e: Check if a year is leap year or not

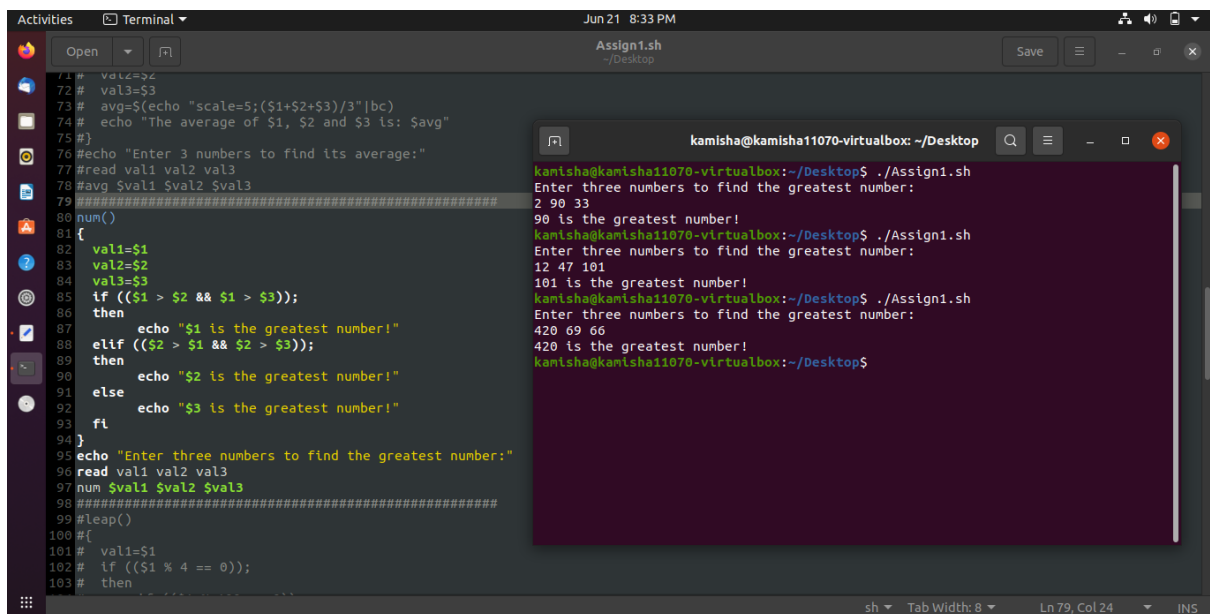


The screenshot shows a terminal window with a file editor open to a script named `Assign1.sh`. The script is a shell script that takes three numbers as input and checks if the first number is a leap year. It uses nested `if` statements to check divisibility by 4, 100, and 400. The script also prompts the user to enter a year to check.

```
95 #echo "Enter three numbers to find the greatest number:"
96 #read val1 val2 val3
97 #num $val1 $val2 $val3
98 #####
99 leap()
100 {
101     val1=$1
102     if ((($1 % 4 == 0));
103     then
104         if ((($1 % 100 == 0));
105         then
106             if ((($1 % 400 == 0));
107             then
108                 echo "This year is a Leap Year!"
109             elif ((($1 % 400 != 0));
110             then
111                 echo "This year is not a Leap Year!"
112             fi
113         elif ((($1 % 100 != 0));
114         then
115             echo "This year is a Leap Year!"
116         fi
117     elif ((($1 % 4 != 0));
118     then
119         echo "This year is not a Leap Year!"
120     fi
121 }
122 #echo "Enter a year to check whether it's a leap year or not:"
123 #read val1
124 #leap $val1
125 #####
126 #even()
127 #if
```

The terminal output shows the script being executed. The user enters 2002, and the script outputs "This year is not a Leap Year!". The user then enters 2016, and the script outputs "This year is a Leap Year!".

## Part – f: Find the largest number among three numbers

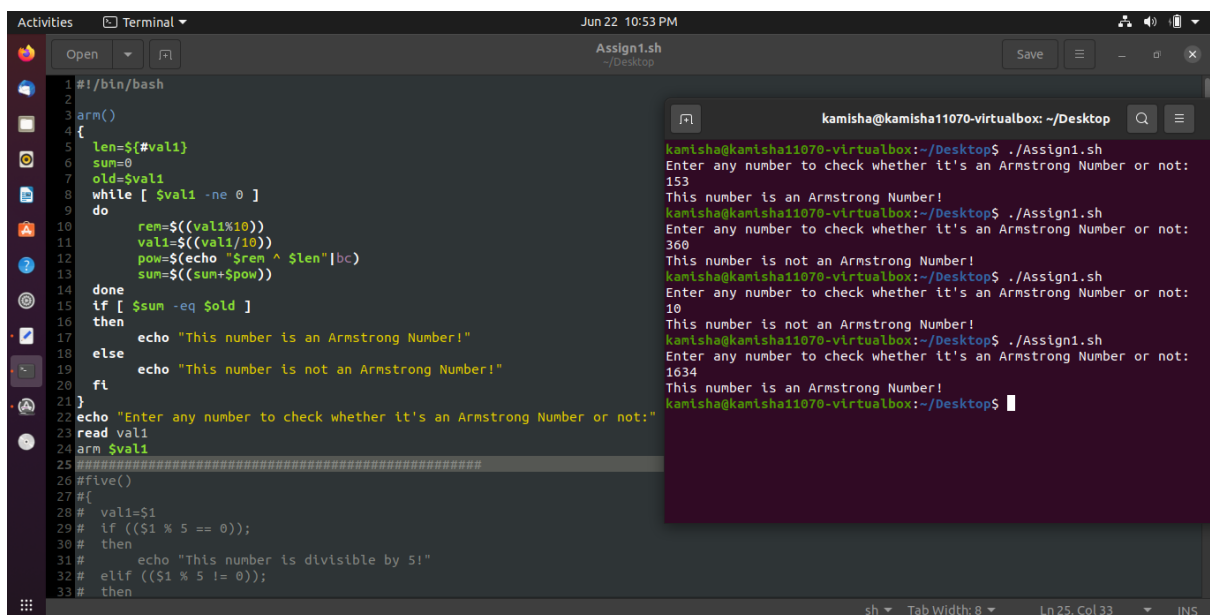


The screenshot shows a terminal window with a file editor open to a script named `Assign1.sh`. The script is a shell script that takes three numbers as input and finds the largest number among them. It uses `if` statements to compare the numbers. The script also prompts the user to enter three numbers to find the greatest number.

```
71 # val1=$1
72 # val2=$2
73 # val3=$3
74 # avg=$((($1+$2+$3)/3))
75 # echo "The average of $1, $2 and $3 is: $avg"
76 #}
77 #echo "Enter 3 numbers to find its average:"
78 #read val1 val2 val3
79 #avg $val1 $val2 $val3
80 #####
81 num()
82 {
83     val1=$1
84     val2=$2
85     val3=$3
86     if ((($1 > $2 && $1 > $3));
87     then
88         echo "$1 is the greatest number!"
89     elif ((($2 > $1 && $2 > $3));
90     then
91         echo "$2 is the greatest number!"
92     else
93         echo "$3 is the greatest number!"
94     fi
95 }
96 #echo "Enter three numbers to find the greatest number:"
97 #read val1 val2 val3
98 #num $val1 $val2 $val3
99 #####
100 #leap()
101 {
102     val1=$1
103     if ((($1 % 4 == 0));
104     then
```

The terminal output shows the script being executed. The user enters 2 90 33, and the script outputs "90 is the greatest number!". The user then enters 12 47 101, and the script outputs "101 is the greatest number!". The user then enters 420 69 66, and the script outputs "420 is the greatest number!".

## Part – g: Check if a number is an Armstrong number or not



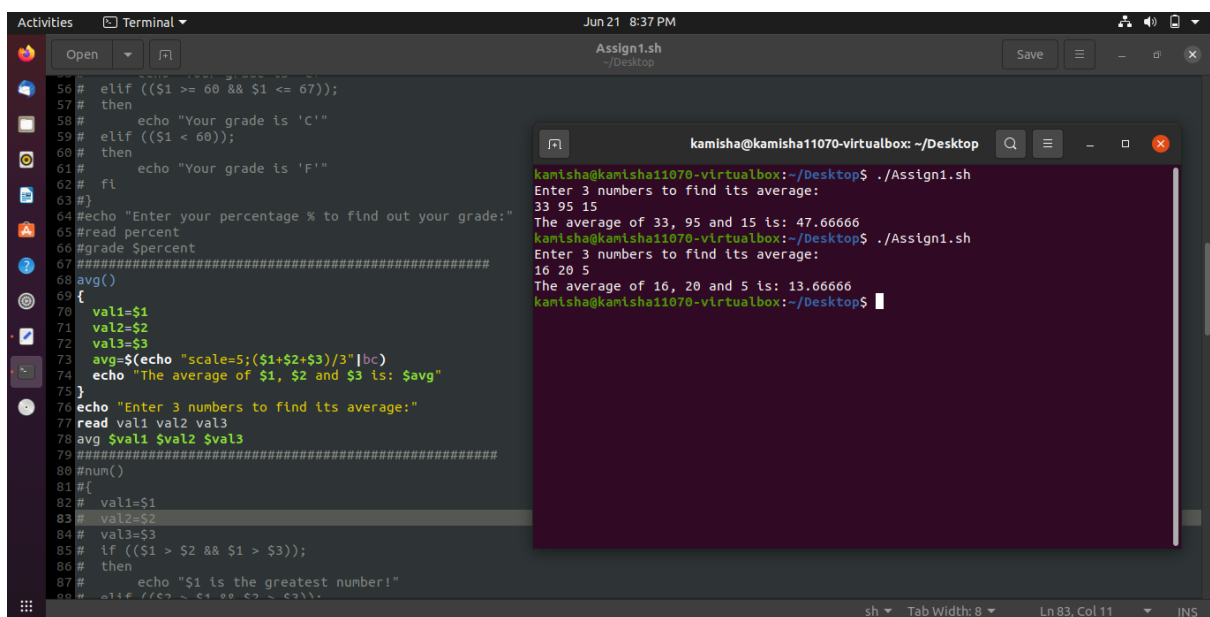
The screenshot shows a terminal window with a file editor open to a script named `Assign1.sh`. The script is a shell script that checks if a number is an Armstrong number. It uses a `while` loop to process the number digit by digit, calculating the sum of the cubes of its digits. The script prompts the user to enter a number and then prints the result.

```
1#!/bin/bash
2
3arm()
4{
5    len=${#val1}
6    sum=0
7    old=$val1
8    while [ $val1 -ne 0 ]
9    do
10        rem=$((val1%10))
11        val1=$((val1/10))
12        pow=$((echo "$rem ^ $len"|bc))
13        sum=$((sum+$pow))
14    done
15    if [ $sum -eq $old ]
16    then
17        echo "This number is an Armstrong Number!"
18    else
19        echo "This number is not an Armstrong Number!"
20    fi
21}
22echo "Enter any number to check whether it's an Armstrong Number or not:"
23read val1
24arm $val1
25#####
26#five()
27#{
28#    val1=$1
29#    if (($1 % 5 == 0));
30#    then
31#        echo "This number is divisible by 5!"
32#    elif (($1 % 5 != 0));
33#    then
```

The terminal output shows the script being executed with the following inputs and outputs:

```
kamisha@kamisha11070-virtualbox: ~/Desktop$ ./Assign1.sh
Enter any number to check whether it's an Armstrong Number or not:
153
This number is an Armstrong Number!
kamisha@kamisha11070-virtualbox: ~/Desktop$ ./Assign1.sh
Enter any number to check whether it's an Armstrong Number or not:
360
This number is not an Armstrong Number!
kamisha@kamisha11070-virtualbox: ~/Desktop$ ./Assign1.sh
Enter any number to check whether it's an Armstrong Number or not:
10
This number is not an Armstrong Number!
kamisha@kamisha11070-virtualbox: ~/Desktop$ ./Assign1.sh
Enter any number to check whether it's an Armstrong Number or not:
1634
This number is an Armstrong Number!
kamisha@kamisha11070-virtualbox: ~/Desktop$
```

## Part – h: Find the average of three values



The screenshot shows a terminal window with a file editor open to a script named `Assign1.sh`. The script is a shell script that finds the average of three values. It prompts the user to enter three numbers and then calculates the average using the formula  $\text{avg} = \frac{\text{val1} + \text{val2} + \text{val3}}{3}$ . The script prints the result.

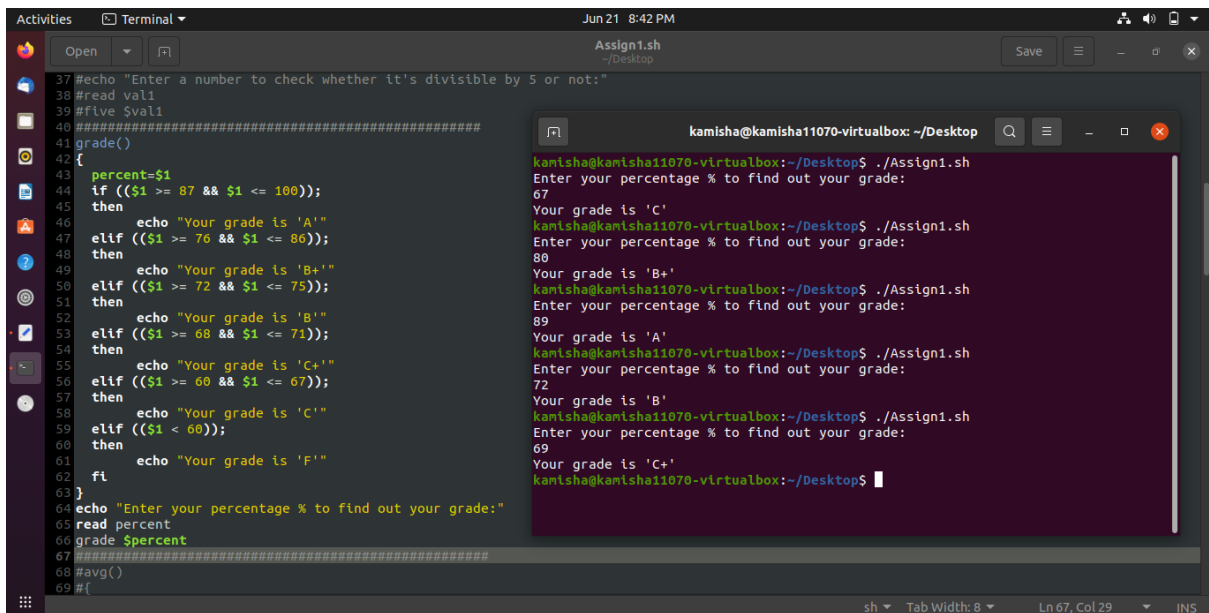
```
56# elif (($1 >= 60 && $1 <= 67));
57# then
58#     echo "Your grade is 'C'"
59# elif (($1 < 60));
60# then
61#     echo "Your grade is 'F'"
62# fi
63#}
64#echo "Enter your percentage % to find out your grade:"
65#read percent
66#grade $percent
67#####
68avg()
69{
70    val1=$1
71    val2=$2
72    val3=$3
73    avg=$((echo "scale=5;($1+$2+$3)/3"|bc))
74    echo "The average of $1, $2 and $3 is: $avg"
75}
76echo "Enter 3 numbers to find its average:"
77read val1 val2 val3
78avg $val1 $val2 $val3
79#####
80#num()
81#{
82#    val1=$1
83#    val2=$2
84#    val3=$3
85#    if (($1 > $2 && $1 > $3));
86#    then
87#        echo "$1 is the greatest number!"
88#    elif (($2 > $1 && $2 > $3));
89#    then
```

The terminal output shows the script being executed with the following inputs and outputs:

```
kamisha@kamisha11070-virtualbox: ~/Desktop$ ./Assign1.sh
Enter 3 numbers to find its average:
33 95 15
The average of 33, 95 and 15 is: 47.66666
kamisha@kamisha11070-virtualbox: ~/Desktop$ ./Assign1.sh
Enter 3 numbers to find its average:
16 20 5
The average of 16, 20 and 5 is: 13.66666
kamisha@kamisha11070-virtualbox: ~/Desktop$
```



## Part – i: Calculate grade 'A' to 'F' after inputting percentage

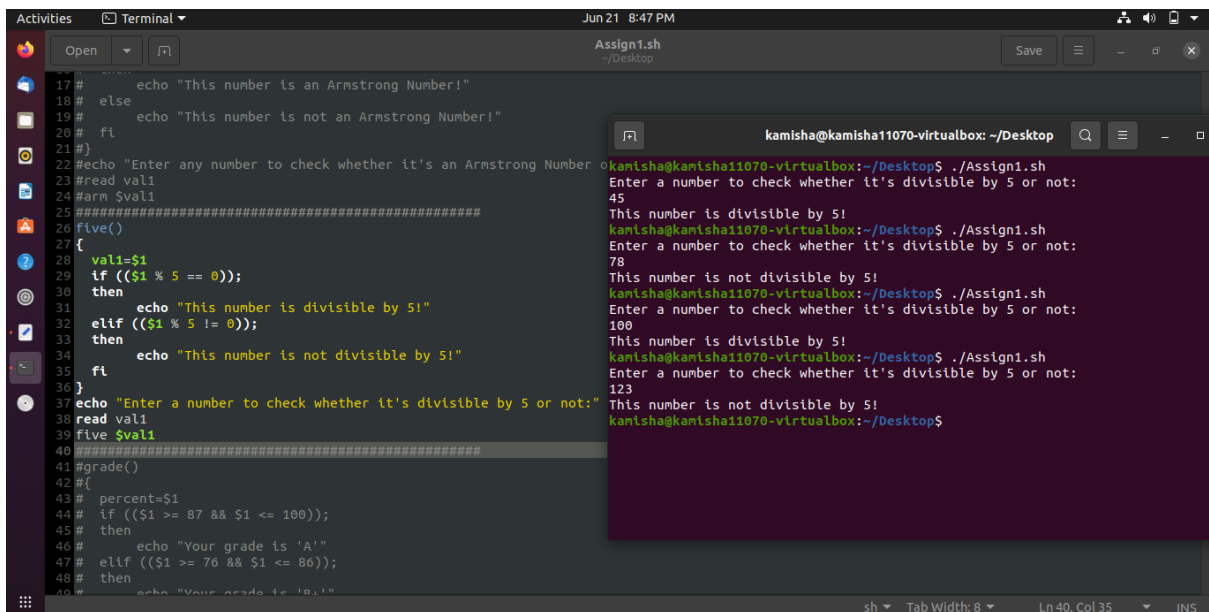


```
37 #echo "Enter a number to check whether it's divisible by 5 or not:"
38 #read val1
39 #five $val1
40 #####
41 grade()
42 {
43     percent=$1
44     if (( $1 >= 87 && $1 <= 100 ));
45     then
46         echo "Your grade is 'A'"
47     elif (( $1 >= 76 && $1 <= 86 ));
48     then
49         echo "Your grade is 'B+'"
50     elif (( $1 >= 72 && $1 <= 75 ));
51     then
52         echo "Your grade is 'B'"
53     elif (( $1 >= 68 && $1 <= 71 ));
54     then
55         echo "Your grade is 'C+'"
56     elif (( $1 >= 60 && $1 <= 67 ));
57     then
58         echo "Your grade is 'C'"
59     elif (( $1 < 60 ));
60     then
61         echo "Your grade is 'F'"
62     fi
63 }
64 #echo "Enter your percentage % to find out your grade:"
65 read percent
66 grade $percent
67 #####
68 #avg()
69 #{
```

Terminal output:

```
kamisha@kamisha11070-virtualbox: ~/Desktop
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter your percentage % to find out your grade:
67
Your grade is 'C'
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter your percentage % to find out your grade:
80
Your grade is 'B+'
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter your percentage % to find out your grade:
89
Your grade is 'A'
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter your percentage % to find out your grade:
72
Your grade is 'B'
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter your percentage % to find out your grade:
69
Your grade is 'C+'
kamisha@kamisha11070-virtualbox:~/Desktop$
```

## Part – j: Check if a value is divisible by 5 or not



```
17 # echo "This number is an Armstrong Number!"
18 # else
19 # echo "This number is not an Armstrong Number!"
20 # fi
21 #}
22 #echo "Enter any number to check whether it's an Armstrong Number or not:"
23 #read val1
24 #arm $val1
25 #####
26 five()
27 {
28     val1=$1
29     if (( $1 % 5 == 0 ));
30     then
31         echo "This number is divisible by 5!"
32     elif (( $1 % 5 != 0 ));
33     then
34         echo "This number is not divisible by 5!"
35     fi
36 }
37 #echo "Enter a number to check whether it's divisible by 5 or not:"
38 read val1
39 five $val1
40 #####
41 #grade()
42 {
43     percent=$1
44     if (( $1 >= 87 && $1 <= 100 ));
45     then
46         echo "Your grade is 'A'"
47     elif (( $1 >= 76 && $1 <= 86 ));
48     then
49         echo "Your grade is 'B+'"
50     elif (( $1 >= 72 && $1 <= 75 ));
51     then
52         echo "Your grade is 'B'"
53     elif (( $1 >= 68 && $1 <= 71 ));
54     then
55         echo "Your grade is 'C+'"
56     elif (( $1 >= 60 && $1 <= 67 ));
57     then
58         echo "Your grade is 'C'"
59     elif (( $1 < 60 ));
60     then
61         echo "Your grade is 'F'"
62     fi
63 }
```

Terminal output:

```
kamisha@kamisha11070-virtualbox: ~/Desktop
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter a number to check whether it's divisible by 5 or not:
45
This number is divisible by 5!
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter a number to check whether it's divisible by 5 or not:
78
This number is not divisible by 5!
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter a number to check whether it's divisible by 5 or not:
100
This number is divisible by 5!
kamisha@kamisha11070-virtualbox:~/Desktop$ ./Assign1.sh
Enter a number to check whether it's divisible by 5 or not:
123
This number is not divisible by 5!
kamisha@kamisha11070-virtualbox:~/Desktop$
```