

## **INDIAN INSTITUTE OF TECHNOLOGY (INDIAN SCHOOL OF MINES) DHANBAD**



**Department of Computer Science and Engineering Operating System Lab  
(CSC15202)**  
**Monsoon Semester (2020-21)**

**Name:** Vivek Kumar Verma  
**Admission Number:** 18JE0940

## **Index**

<u><b>Exp No</b></u>	<u><b>Aim</b></u>	<u><b>Date</b></u>	<u><b>Signature</b></u>	<u><b>Remarks</b></u>
<b>1</b>	Basics of UNIX Commands.	August 8, 2020		
<b>2</b>	Shell Programming	August 18, 2020		
<b>3</b>	Implementation of CPU Scheduling. A) FCFS B) SJF	August 25, 2020		
<b>4</b>	Priority Scheduling	September 1, 2020		
<b>5</b>	Round Robin Scheduling	September 1, 2020		
<b>6</b>	Implementation of SJF (Preemptive and Non-preemptive)	September 8, 2020		
<b>7</b>	Multilevel Queue Scheduling Algorithm - I	September 22, 2020		
<b>8</b>	Multilevel Queue Scheduling Algorithm - II	September 22, 2020		
<b>9</b>	Producer & Consumer Algorithm	September 29, 2020		
<b>10</b>	Dining Philosophers Problem	October 6, 2020		
<b>11</b>	Banker's and Deadlock Prevention Algorithm	October 13, 2020		
<b>12</b>	Simulate Page Replacement Algorithms: FIFO	October 20, 2020		
<b>13</b>	Simulate Page Replacement Algorithms: LRU	October 20, 2020		

14	Simulate Fixed Partitioning: First Fit, Best Fit and Worst Fit	October 27, 2020		
15	Simulate Partitioning: First Fit, Best Fit and Worst Fit	October 27, 2020		

## Experiment Number 1: Basics of UNIX commands

**Date:**August 11, 2020

**Aim:**To write and learn the basic Linux commands.

<b><u>Linux Command</u></b>	<b><u>DOS Command</u></b>	<b><u>Description</u></b>
<b>pwd</b>	cd	“Print WorkingDirectory”. Shows thecurrent location in the directory tree.
<b>cd</b>	cd, chdir	“Change Directory”. When typed all by itself, it returns you to your home directory.
<b>cd directory</b>	cd directory	Change into the specified directoryname. Example: cd /usr/src/linux
<b>cd ~</b>		“~” is an alias for your home directory. It can be used as a shortcut to your “home”, or other directories relative to your home.
<b>cd..</b>	cd..	Move up one directory. For example, if you are in /home/vic and you type “cd ..”, you will end up in /home.
<b>cd -</b>		Return to previous directory. An easy way to get back to your previous location!
<b>ls</b>	dir /w	List all files in the current directory, in column format.
<b>ls directory</b>	dir directory	List the files in the specified directory. Example: ls /var/log
<b>ls -l</b>	dir	List files in “long” format, one file per line. This also shows you additional info about the file, such as ownership, permissions, date, and size.
<b>ls -a</b>	dir /a	List all files, including “hidden” files. Hidden files are those files that begin with a “.”, e.g. The .bash_history file in your home directory.
<b>ls -ld directory</b>		A “long” list of “directory”, but instead of showing the directory contents, show the directory's detailed information. For example, compare the output of the following two commands:  ls -l /usr/bin ls -ld /usr/bin
<b>ls /usr/bin/d*</b>	dir d*.*	List all files whose names begin with the letter “d” in the /usr/bin directory.

<i><b>Linux Command</b></i>	<i><b>DOS Command</b></i>	<i><b>Description</b></i>
<b>pwd</b>	cd	“Print WorkingDirectory”. Shows thecurrent location in the directory tree.
<b>cd</b>	cd, chdir	“Change Directory”. When typed all by itself, it returns you to your home directory.
<b>cd directory</b>	cd directory	Change into the specified directoryname. Example: cd /usr/src/linux
<b>cd ~</b>		“~” is an alias for your home directory. It can be used as a shortcut to your “home”, or other directories relative to your home.
<b>cd..</b>	cd..	Move up one directory. For example, if you are in /home/vic and you type “cd ..”, you will end up in /home.
<b>cd -</b>		Return to previous directory. An easy way to get back to your previous location!
<b>ls</b>	dir /w	List all files in the current directory, in column format.
<b>ls directory</b>	dir directory	List the files in the specified directory. Example: ls /var/log
<b>ls -l</b>	dir	List files in “long” format, one file per line. This also shows you additional info about the file, such as ownership, permissions, date, and size.
<b>ls -a</b>	dir /a	List all files, including “hidden” files. Hidden files are those files that begin with a “.”, e.g.The .bash_history file in your home directory.
<b>ls -ld directory</b>		A “long” list of “directory”, but instead of showing the directory contents, show the directory's detailed information. For example, compare the output of the following two commands: ls -l /usr/bin ls -ld /usr/bin
<b>ls /usr/bin/d*</b>	dir d*.*	List all files whose names begin with the letter “d” in the /usr/bin directory.

## Experiment Number 2: Shell programming

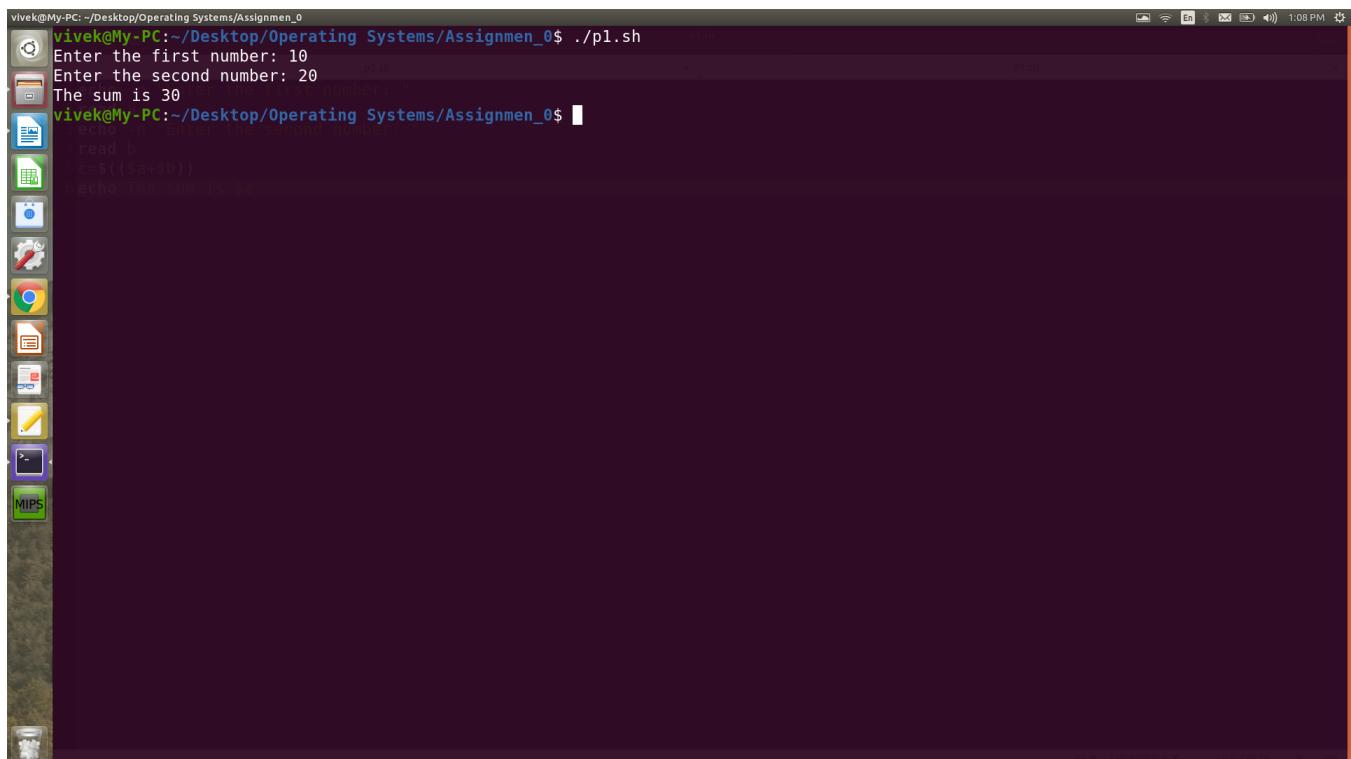
**Date:** August 18, 2020

**Aim:** To write the given programs to implement Shell Programming.

**Q1. Write a shell program to add two numbers.**

**Sol:**

```
echo -n "Enter the first number: "
read a
echo -n "Enter the second number: "
read b
c=$((a+b))
echo The sum is $c
```



The screenshot shows a terminal window titled 'vivek@My-PC: ~/Desktop/Operating Systems/Assignmen\_0\$'. The user has run the command './p1.sh'. The terminal displays the following interaction:

```
vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$ ./p1.sh
Enter the first number: 10
Enter the second number: 20
The sum is 30
vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$
```

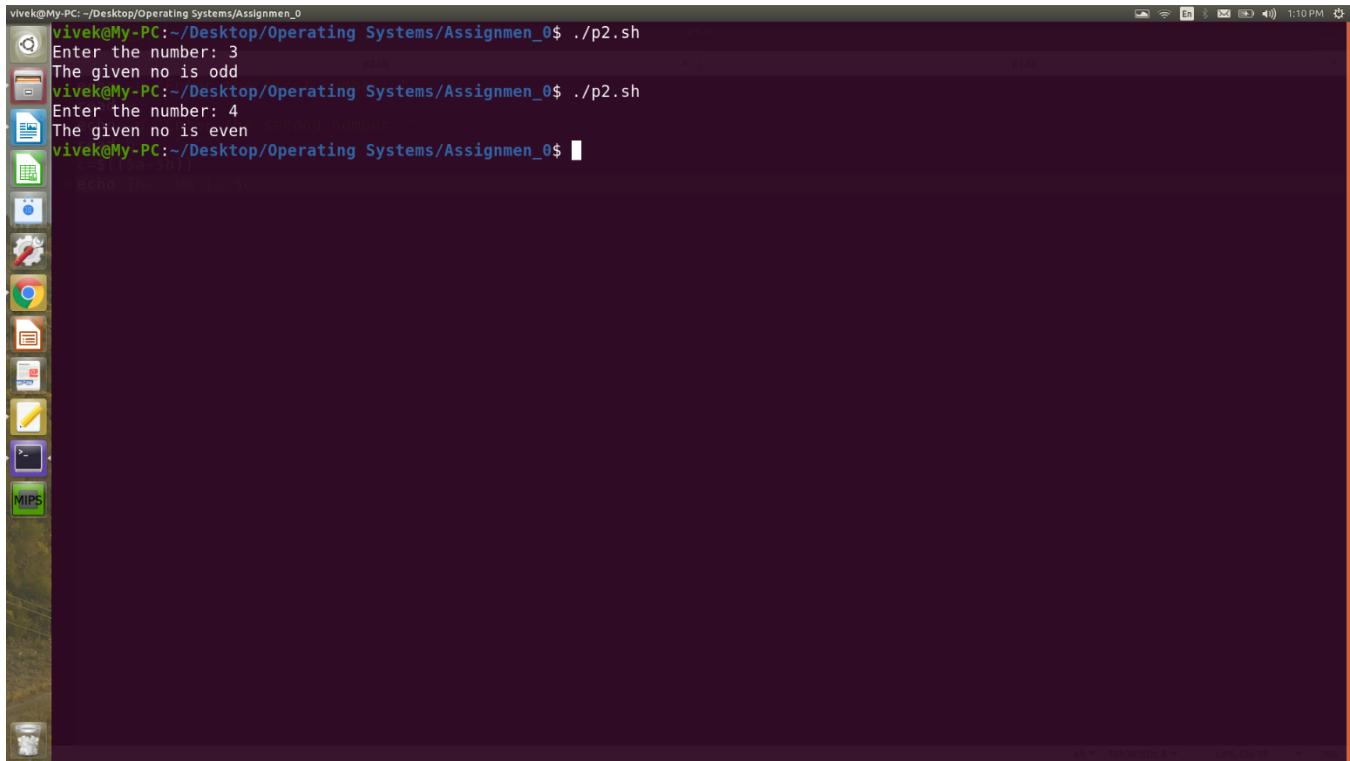
**Q2 Write a shell program to find whether a number is even or odd.**

**Sol:**

```
echo -n "Enter the number: "
read x
b=$((x%2))
if [ $b -eq 0 ]; then
echo The given no is even
```

```
else
echo The given no is odd
fi
```

## Output



```
vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$ ./p2.sh
Enter the number: 3
The given no is odd
vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$ ./p2.sh
Enter the number: 4
The given no is even
```

## **Q3 Write a shell program for Fibonacci series.**

**Sol.**

```
count=1
f1=1
f2=1
k=0
n=0
echo -n "Enter the value of n: "
read n
if [ $n -ge 1 ]
then
    echo $f1
fi

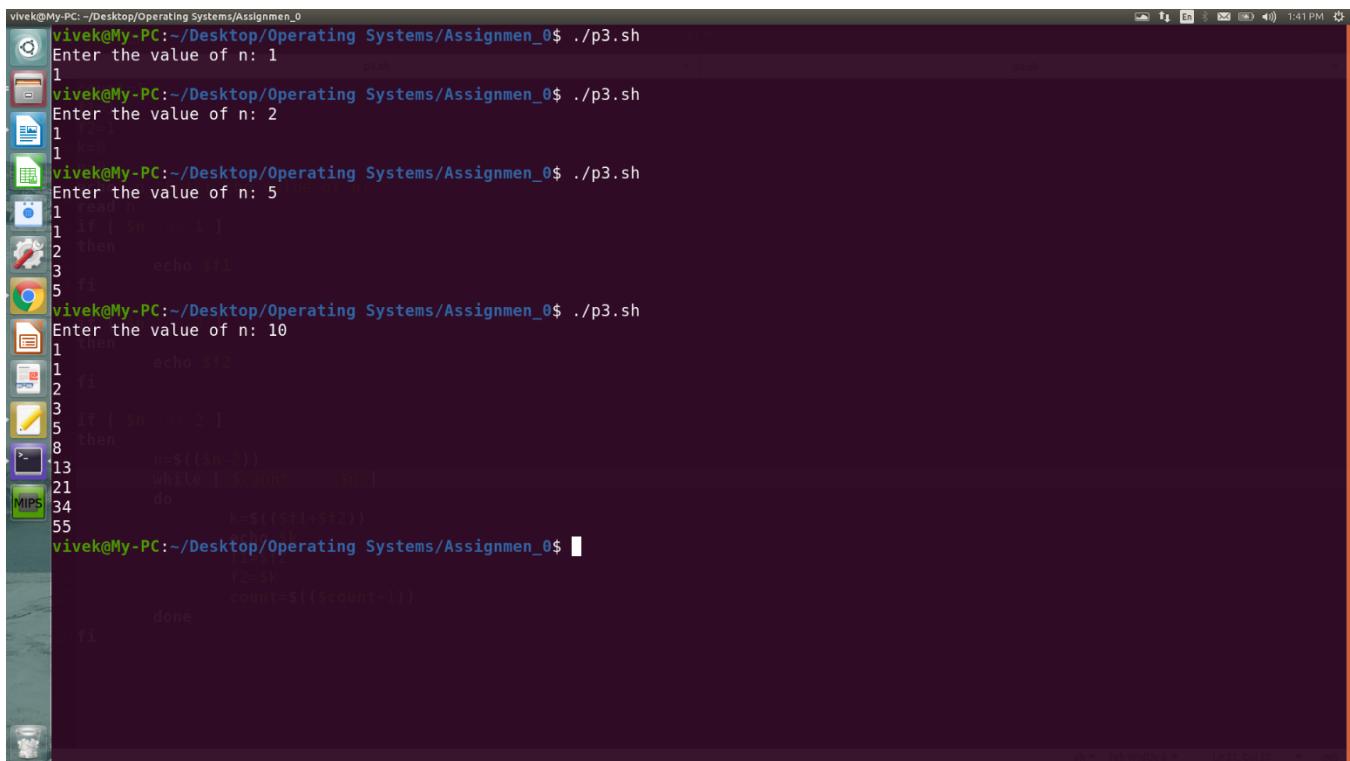
if [ $n -ge 2 ]
then
    echo $f2
fi
```

```

if [ $n -gt 2 ]
then
    while [ $count -le $($n-2) ]
    do
        k=$((f1+f2))
        echo $k
        f1=$f2
        f2=$k
        count=$((count+1))
    done
fi

```

## Output



```

vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$ ./p3.sh
Enter the value of n: 1
1
vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$ ./p3.sh
Enter the value of n: 2
1
1
vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$ ./p3.sh
Enter the value of n: 5
1
read
1 if [ $n -gt 1 ]
1 then
2     echo $f1
3     fi
5 fi
vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$ ./p3.sh
Enter the value of n: 10
1
1
2
3
4
5 if [ $n -gt 2 ]
5 then
6     n=$((n-2))
7     while [ $count -le $n ]
8     do
9         k=$((f1+f2))
10        echo $k
11        f1=$f2
12        f2=$k
13        count=$((count+1))
14    done
15
16    fi
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
vivek@My-PC:~/Desktop/Operating Systems/Assignmen_0$ 

```

## Experiment Number 3: Implementation of CPU Scheduling.

Date:August 25, 2020

Aim:To write the program to implement following CPU scheduling algorithms.

1. FCFS
2. SJF

### 1. Algorithm for FCFS

1.	Start the program.
2.	Get the number of processes, arrival time and burst times. (also ID's if required)
3.	Compute the response time, completion time, waiting time and turnaround time.
4.	The waiting time of all the processes is summed then average value time is calculated.
5.	The waiting time of each process and average times are displayed
6.	Same is done for turnaround time as well.
7.	Stop the program

### C++ implementation

```
#include<bits/stdc++.h>

using namespace std;

struct process
{
    int at,bt,wt,ct,tat;
    string id;

    process()
    {
        id = "";
        at = bt = wt = ct = tat = -1;
    }
};

bool compare(process p1, process p2)
{
    if(p1.at==p2.at)
    {
        return p1.id < p2.id;
    }
    return p1.at < p2.at;
}

int main()
```

```

{
    int n,i;
    cout<<"Enter the number of processes: ";
    cin>>n;
    process p1[n];
    cout<<"Enter the id, arrival time and burst time separated by spaces:-"<<endl;
    for(i=0;i<n;++i)
    {
        cin>>p1[i].id>>p1[i].at>>p1[i].bt;
    }
    sort(p1,p1+n,compare);
    int time = 0;
    float tat_av = 0, wat_avg = 0;
    for(i=0;i<n;++i)
    {
        time += p1[i].bt;
        p1[i].ct = time;
        p1[i].tat = p1[i].ct - p1[i].at;
        p1[i].wt = p1[i].tat - p1[i].bt;
        tat_av += p1[i].tat;
        wat_avg += p1[i].wt;
    }
    cout<<"\nCPU scheduling is as follows :-\n";
    cout<<"PID\tAT\tBT\tCT\tTAT\tWT\n";
    for(i=0;i<n;++i)
    {
        cout<<p1[i].id<<"\t"<<p1[i].at<<"\t"<<p1[i].bt<<"\t"<<p1[i].ct<<"\t"<<p1[i].tat<<"\t"<<
p1[i].wt<<endl;
    }
    cout<<endl;
    cout<<"Average turn around time: "<<tat_av / n<<endl;
    cout<<"Average waiting time: "<<wat_avg / n<<endl;
    return 0;
}

```

## Output

```
vivek@18je0940:~/Desktop/Operating Systems/Assignment 1
vivek@18je0940:~/Desktop/Operating Systems/Assignment 1$ ./a.out
Enter the number of processes: 4
Enter the id, arrival time and burst time separated by spaces:- P1 0 4
P1 0 4
P2 1 3
P3 2 5
P4 3 2
CPU scheduling is as follows :-
PID    AT     BT     CT     TAT     WT
P1     0      4      4      4      0
P2     1      3      7      6      3
P3     2      5     12     10     5
P4     3      2     14     11     9
ARRIVAL TIME          BURST TIME
P3           2           5
P4           3           2
P1           0           4
P2           1           3
P3           2           5
P4           3           2
0           12           4
12          14           7
Avg. TAT= 7.75
Avg. WT= 4.25
```

```
vivek@18je0940:~/Desktop/Operating Systems/Assignment 1
vivek@18je0940:~/Desktop/Operating Systems/Assignment 1$ ./a.out
Enter the number of processes: 7
Enter the id, arrival time and burst time separated by spaces:- P1 0 1
P1 0 1
P2 1 6
P3 1 3
P4 4 5
P5 2 8
P6 6 10
P7 6 3
CPU scheduling is as follows :-
PID    AT     BT     CT     TAT     WT
P1     0      1      1      1      0
P2     1      6      7      6      0
P3     1      3     10     9      6
P5     2      8     18     16     8
P4     4      5     23     19     14
P6     6     10     33     27    17
P7     6      3     36     30     27
P1           0           1
P2           1           6
P3           1           3
P5           8           6
P4           14          10
P6           1           6
P7           3           3
Avg. turn around time: 15.4286
Avg. waiting time: 10.2857
vivek@18je0940:~/Desktop/Operating Systems/Assignment 1$
```

vivek@18je0940:~/Desktop/Operating Systems/Assignment 1

```
vivek@18je0940:~/Desktop/Operating Systems/Assignment 1$ ./a.out
Enter the number of processes: 5
Enter the id, arrival time and burst time separated by spaces:- P1 2 6
P2 5 2
P3 1 8
P4 0 3
P5 4 4
CPU scheduling is as follows :-
PID    AT      BT      CT      TAT      WT
P4     0       3       3       3       0
P3     1       8       11      10      2
P1     2       6       17      15      9
P5     4       4       21      17      13
P2     5       2       23      18      16
Average turn around time: 12.6
Average waiting time: 8
vivek@18je0940:~/Desktop/Operating Systems/Assignment 1$
```

**EXAMPLE 3**

Write the programs for both the algorithm and show the output of example 2 and example 3 for both the code.

You have to send the code, screenshots of the output screen. Please mentioned your name, admission number in the mail. Submit it by today midnight.

The mail id is- oslab2020@gmail.com

## 2. Algorithm for SJF (non-preemptive)

1.	Start the program.
2.	Get the number of processes, arrival time and burst times. (also ID's if required)
3.	Compute the response time, completion time, waiting time and turnaround time.
4.	The waiting time of all the processes is summed then average value time is calculated.
5.	The waiting time of each process and average times are displayed
6.	Same is done for turnaround time as well.
7.	Stop the program

### C++ implementation

```
#include<bits/stdc++.h>

using namespace std;

struct job
{
    int at, bt, st, ct, tat, wt;
    string id;
};

void non_preemptive(job j1[], float &avg1, float &avg2, int n)
{
    avg1 = 0;
    avg2 = 0;
    int rem[n],i;
    for(i=0;i<n;++i)
    {
        rem[i] = j1[i].bt;
    }
    int completed = 0,time = 0;
    while(completed<n)
    {
        int min_time = INT_MAX,min_index = -1;
        for(i=0;i<n;++i)
        {
            if(j1[i].at<=time and rem[i]>0 and rem[i]<min_time)
            {
                min_time = rem[i];
                min_index = i;
            }
        }
        if(min_index == -1)
        {
```

```

        time = time + 1;
        continue;
    }
    j1[min_index].st = time;
    j1[min_index].ct = time + j1[min_index].bt;
    rem[min_index] = 0;

    time = time + j1[min_index].bt;

    min_time = rem[min_index];

    completed += 1;
    min_time = INT_MAX;

    j1[min_index].tat = j1[min_index].ct - j1[min_index].at;
    j1[min_index].wt = j1[min_index].tat - j1[min_index].bt;

    avg1 += j1[min_index].tat;
    avg2 += j1[min_index].wt;

    //time = time + 1;
}
avg1 = avg1 / n;
avg2 = avg2 / n;
cout<<"ID\tAT\tBT\tST\tCT\tWT\tTAT"<<endl;
for(i=0;i<n;++i)
{
    cout<<j1[i].id<<"\t"<<j1[i].at<<"\t"<<j1[i].bt<<"\t"<<j1[i].st<<"\t"<<j1[i].ct<<"\t"<<j1[i].wt<<"\t"<<j1[i].tat<<endl;
}
}

int main()
{
    int n,i;
    cout<<"Enter the number of processes: ";
    cin>>n;
    job j1[n];
    cout<<"Enter the id, arrival time and burst time separated by spaces:-"<<endl;
    for(i=0;i<n;++i)
    {
        cin>>j1[i].id>>j1[i].at>>j1[i].bt;
    }
    float avg1,avg2;
    cout<<"\nThe job schedule is as follows :-\n";
    non_preemptive(j1,avg1,avg2,n);
}

```

```

        cout<<"Average turn around time: "<<avg1<<endl;
        cout<<"Average waiting time: "<<avg2<<endl;
        return 0;
    }
}

```

## Output

```

vivek@18je0940:~/Desktop/Operating Systems/Assignment 1$ ./a.out
Enter the number of processes: 4
Enter the id, arrival time and burst time separated by spaces:-
P1 0 4
P2 1 3
P3 2 5
P4 3 2
CPU scheduling is as follows :-
PID   AT     BT      CT      TAT      WT
P1    0      4       4      4       0
P2    1      3       9      8       5
P3    2      5      14     12      7
P4    3      2       6      3       1
Average turn around time: 6.75
Average waiting time: 3.25
vivek@18je0940:~/Desktop/Operating Systems/Assignment 1$ 

```

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
220
```

```

vivek@18je0940:~/Desktop/Operating Systems/Assignment_1$ vivek@18je0940:~/Desktop/Operating Systems/Assignment_1$ ./a.out
Enter the number of processes: 7
Enter the id, arrival time and burst time separated by spaces:-
P1 0 1
P2 1 6
P3 1 3
P4 4 5
P5 2 8
P6 6 10
P7 6 3
CPU scheduling is as follows :-
PID    AT     BT     CT     TAT     WT
P1     0      1      1      1      0
P2     1      6     18     17     11
P3     1      3      4      3      0
P4     4      5      9      5      0
P5     2      8     26     24     16
P6     6     10     36     30     20
P7     6      3     12      6      3
Average turn around time: 12.2857
Average waiting time: 7.14286
vivek@18je0940:~/Desktop/Operating Systems/Assignment_1$ cat main.cpp
#include <iostream>
using namespace std;
int main()
{
    float avg1 = 0, avg2 = 0;
    cout<<"Enter the number of processes: ";
    cin>>n;
    process pl[n];
    map <int, vector <int>> mpl;
    cout<<"Enter the id, arrival time and burst time separated by spaces: "<<endl;
    for(i=0;i<n;i++)
    {
        cin>>pl[i].id>>pl[i].at>>pl[i].bt;
    }
    find(pl,n,avg1,avg2);
    cout<<"\nCPU scheduling is as follows :-\n";
    cout<<"PID\tAT\tBT\tCT\tTAT\tWT\n";
    for(i=0;i<n;i++)
    {
        cout<<pl[i].id<<"\t";
        cout<<pl[i].at<<"\t";
        cout<<pl[i].bt<<"\t";
        cout<<pl[i].ct<<"\t";
        cout<<pl[i].tat<<"\t";
        cout<<pl[i].wt<<endl;
    }
}

```

```

vivek@My-PC:~/Desktop/Operating Systems/Assignment_1$ vivek@My-PC:~/Desktop/Operating Systems/Assignment_1$ ./a.out
Enter the number of processes: 5
Enter the id, arrival time and burst time separated by spaces:-
P1 2 6
P2 5 2
P3 1 8
P4 0 3
P5 4 4
The job schedule is as follows :-
ID    AT     BT     ST     CT     WT     TAT
P1    2      6      3      9      1      7
P2    5      2      9     11      4      6
P3    1      8     15     23     14     22
P4    0      3      0      3      0      3
P5    4      4     11     15      7     11
Average turn around time: 9.8
Average waiting time: 5.2
vivek@My-PC:~/Desktop/Operating Systems/Assignment_1$ 

```

**Result:** The following CPU scheduling algorithms were implemented successfully.

1. First Come First Serve
2. Shortest Job First (Non-preemptive)

## **Experiment Number 4: Implementation of priority scheduling.**

**Date:** September 1, 2020

**Aim:** To write the program to implement priority CPU scheduling.

### **Algorithm for Priority Scheduling**

1.	Start the program.
2.	Get the number of processes, arrival time, burst times and priority. (also ID's if required)
3.	Compute the response time, completion time, waiting time and turnaround time based on priorities.
4.	The waiting time of all the processes is summed then average value time is calculated.
5.	The waiting time of each process and average times are displayed
6.	Same is done for turn around time as well.
7.	Stop the program

### **C++ implementation**

```
#include<bits/stdc++.h>

using namespace std;

struct process
{
    int id,at,bt,priority,st,ft,tat,wt;
};

int main()
{
    int n,i;
    cout<<"Enter the number of processes: ";
    cin>>n;
    process p1[n];
    int remaining[n];
    cout<<"Enter arrival time, burst time and priority :-\n";
    for(i=0;i<n;++i)
    {
        cin>>p1[i].at>>p1[i].bt>>p1[i].priority;
        p1[i].id = i + 1;
        remaining[i] = p1[i].bt;
    }
    int completed = 0,time = 0;
    bool visited[n] = { };
    float avg1 = 0,avg2 = 0;
    int t1 = INT_MAX,t2 = INT_MIN;
```

```

while(completed<n)
{
    int min1 = INT_MIN,idx = -1;
    for(i=0;i<n;++i)
    {
        if(p1[i].priority > min1 and remaining[i]>0 and p1[i].at <= time)
        {
            min1 = p1[i].priority;
            idx = i;
        }
        if(p1[i].priority==min1 and remaining[i]>0 and p1[i].at <= time)
        {
            if(p1[idx].at>p1[i].at)
            {
                idx = i;
            }
        }
    }
    if(idx===-1)
    {
        time += 1;
        continue;
    }
    if(visited[idx]==false)
    {
        p1[idx].st = time;
        visited[idx] = true;
    }
    time += 1;
    remaining[idx] -= 1;
    if(remaining[idx]==0)
    {
        completed += 1;
        p1[idx].ft = time;
        p1[idx].tat = p1[idx].ft - p1[idx].at;
        p1[idx].wt = p1[idx].tat - p1[idx].bt;
        avg1 += p1[idx].tat;
        avg2 += p1[idx].wt;
        t1 = min(t1,p1[idx].at);
        t2 = max(t2,p1[idx].ft);
    }
}
cout<<"\nID\tAT\tPTY\tBT\tST\tFT\tTAT\tWT\n";
for(i=0;i<n;++i)
{
    cout<<p1[i].id<<"\t"<<p1[i].at<<"\t"<<p1[i].priority<<"\t"<<p1[i].bt<<"\t"<<p1[i].st<<"\t"<<p1[i].ft<<"\t"<<p1[i].tat<<"\t"<<p1[i].wt<<"\t"<<p1[i].wt
}

```

```

        \t" << p1[i].ft << "\t" << p1[i].tat << "\t" << p1[i].wt << "\r" << endl;
    }
    cout << "\nAverage turn around time: " << avg1 / n << endl;
    cout << "Average waiting time: " << avg2 / n << endl;
    cout << "Throughput: " << (float)n / (float)(t2-t1) << endl;
    return 0;
}

```

## Output

```

vivek@My-PC:~/Desktop/Operating Systems/Assignment_2$ ./a.out
Enter the number of processes: 7
Enter arrival time, burst time and priority :-
0 4 2
1 2 4
2 3 6
3 5 10
4 1 8
5 4 12
6 6 9

ID      AT      PTY      BT      ST      Assignent      Q      TAT      WT      Priority      Scheduling
1       0       2       4       0       25      25      21
2       1       4       2       1       22      21      19
3       2       6       3       2       Priority 21 during slot 19 the num 16 higher the priority
4       3       10      5       3       12      9       4
5       4       8       1       18      19      15      14
6       5       12      4       5       9       4       0
7       6       9       6       12      18      20      12      0       6
Average turn around time: 15
Average waiting time: 11.4286
Throughput: 0.28
vivek@My-PC:~/Desktop/Operating Systems/Assignment_2$ 

```

**Result:** Priority CPU scheduling algorithm was implemented and verified successfully

## **Experiment Number 5: Implementation of Round Robin Scheduling**

**Date:**September 1, 2020

**Aim:**To write the program to implement round robin scheduling.

### **Algorithm for Priority Scheduling**

1.	Get the number of process and their burst time and two different quantum times and do steps 2 through 6 for both.
2.	Initialize the array for Round Robin circular queue as '0' in both cases.
3.	The burst time of each process is divided and the quotients are stored on the round Robinarray.
4.	According to the array value the waiting time for each process and the average time are calculated as like the otherscheduling.
5.	The waiting time for each process and average times are displayed.
6.	Same is done for turnaround time as well.
7.	Compare the average waiting time, turnaround time and throughput for both time quantums.
8.	Stop the program

### **C++ Implementation**

```
#include<bits/stdc++.h>

using namespace std;

struct process
{
    int at,bt,tat,wt,ct,rt,id;
    process()
    {
        at = bt = tat = wt = ct = rt = id;
    }
};

void update_queue(queue <int>&q1, process p1[], int time, int n, bool pushed[])
{
    int i;
    for(i=0;i<n;++i)
    {
        if(p1[i].at<=time and pushed[i]==false)
        {
            q1.push(p1[i].id);
            pushed[i] = true;
        }
    }
}
```

```

void round_robin(process p1[], int n, int q, float &avg1, float &avg2)
{
    avg1 = 0, avg2 = 0;
    int time1 = 0, completed = 0;
    queue <int> ready;
    int rem[n], i, j;
    bool pushed[n] = { }, visited[n] = { };
    for(i=0; i<n; ++i)
    {
        rem[i] = p1[i].bt;
    }
    update_queue(ready, p1, time1, n, pushed);
    while(completed < n)
    {
        if(ready.empty() == false)
        {
            int temp = ready.front();
            ready.pop();
            if(visited[temp] == false)
            {
                visited[temp] = true;
                p1[temp].rt = time1;
            }
            if(rem[temp] <= q)
            {
                // updating time variable
                time1 = time1 + rem[temp];
                // assigning completion time
                p1[temp].ct = time1;
                // updating remaining time
                rem[temp] = 0;
                // computing turn around time
                p1[temp].tat = p1[temp].ct - p1[temp].at;
                // computing waiting time
                p1[temp].wt = p1[temp].tat - p1[temp].bt;
                // updating complete variable
                completed = completed + 1;
                // computing the averages
                avg1 += p1[temp].tat;
                avg2 += p1[temp].wt;
                // updating ready queue
                update_queue(ready, p1, time1, n, pushed);
            }
            else
            {
                // updating remainning time

```

```

        rem[temp] = rem[temp] - q;
        //updating time variable
        time1 = time1 + q;
        update_queue(ready,p1,time1,n,pushed);
        ready.push(temp);
    }
}
else
{
    time1 += q;
    update_queue(ready,p1,time1,n,pushed);
}
}
avg1 /= n;
avg2 /= n;
cout<<"Time quantum: "<<q<<endl;
cout<<"ID\tAT\tBT\tST\tFT\tTAT\tWT\n";
for(i=0;i<n;++i)
{
    cout<<p1[i].id<<"\t"<<p1[i].at<<"\t"<<p1[i].bt<<"\t"<<p1[i].rt<<"\t"<<p1[i].ct<<"\t"<<
p1[i].tat<<"\t"<<p1[i].wt<<"\r"<<endl;
}
cout<<"Average Turn Around Time: "<<avg1<<endl;
cout<<"Aversge Waiting Time: "<<avg2<<endl;
int t1,t2;
t1 = INT_MAX;
t2 = INT_MIN;
for(i=0;i<n;++i)
{
    t1 = min(p1[i].at,t1);
    t2 = max(p1[i].ct,t2);
}
cout<<"Throughput: "<<1.0 * n / (t2 - t1)<<endl;
}

int main()
{
    int n,q1,q2,i;
    cout<<"Enter the number of processes followed by 2 time quantums: ";
    cin>>n>>q1>>q2;
    process p1[n];
    cout<<"Enter the arrival time and burst time for each process: "<<endl;
    for(i=0;i<n;++i)
    {
        p1[i].id = i;
        cin>>p1[i].at>>p1[i].bt;
    }
}

```

```

float a,b,c,d;
round_robin(p1,n,q1,a,b);
round_robin(p1,n,q2,c,d);
cout<<"The differences are :-\n";
cout<<"Average turn around time: "<<abs(c-a)<<endl;
cout<<"Average turn around time: "<<abs(d-b)<<endl;
return 0;
}

```

## Output

```

vivek@My-PC:/media/vivek/Logical/Operating Systems/Assignment_2$ g++ round_robin.cpp
vivek@My-PC:/media/vivek/Logical/Operating Systems/Assignment_2$ ./a.out
Enter the number of processes followed by 2 time quanta: 6 1 2
Enter the arrival time and burst time for each process:
0 4
1 5
2 2
3 1
4 6
5 3
Time quantum: 1
ID   AT    BT    ST    FT    TAT    WT
0    0     4     0    12    12     8
1    1     5     1    17    16    11
2    2     2     3     9    7     5
3    3     1     5     6    3     2
4    4     6     7    21    17    11
5    6     3    10    18    12    9
Average Turn Around Time: 11.1667
Aversgae Waiting Time: 7.66667
Throughput: 0.285714
Time quantum: 2
ID   AT    BT    ST    FT    TAT    WT
MIPS 0    0     4     0     8     8     4
1    1     5     2    18    17    12
2    2     2     4     6     4     2
3    3     1     8     9     6     5
4    4     6     9    21    17    11
5    6     3    13    19    13    10
Average Turn Around Time: 10.8333
Aversgae Waiting Time: 7.33333
Throughput: 0.285714
The differences are :-
Average turn around time: 0.333334
Average turn around time: 0.333333
vivek@My-PC:/media/vivek/Logical/Operating Systems/Assignment_2$ 

```

**Result:** Round Robin CPU scheduling algorithm was implemented and verified successfully.

## Experiment Number 6: Implementation of SJF (Preemptive & Non-preemptive Scheduling)

Date:September 8, 2020

Aim:To write the program to implement preemptive and non-preemptive scheduling and compare the results of the two.

### Algorithm for Preemptive SJF

1.	Get the number of process and their burst time, arrival time and IDs if needed.
2.	Select the next job from the ready queue which has the shortest remaining time.
3.	If in between a new process arrives with even shorter remaining time, the CPU turns to that process immediately.
4.	Computer waiting time, turnaround time and response time for each process.
5.	The waiting time for each process and average times are displayed.
6.	Same is done for turnaround time as well.
7.	Compute the above quantities using non-preemptive algorithm as well and compare the values.
7.	Stop the program.

### C++ Implementation

```
#include<bits/stdc++.h>

using namespace std;

struct job
{
    int id, at, bt, st, ct, tat, wt;
};

void preemptive(job j1[], float &avg1, float &avg2, int n)
{
    avg1 = 0;
    avg2 = 0;
    int rem[n],i;
    bool start[n] = { };
    for(i=0;i<n;++i)
    {
        rem[i] = j1[i].bt;
    }
    int completed = 0,time = 0;
    while(completed<n)
    {
        int min_time = INT_MAX,min_index = -1;
```

```

for(i=0;i<n;++i)
{
    if(j1[i].at<=time and rem[i]>0 and rem[i]<min_time)
    {
        min_time = rem[i];
        min_index = i;
    }
}
if(min_index== -1)
{
    time = time + 1;
    continue;
}
if(start[min_index]==false)
{
    j1[min_index].st = time;
    start[min_index] = true;
}
rem[min_index] -= 1;
min_time = rem[min_index];
if(min_time == 0)
{
    completed += 1;
    min_time = INT_MAX;

    j1[min_index].ct = time + 1;
    j1[min_index].tat = j1[min_index].ct - j1[min_index].at;
    j1[min_index].wt = j1[min_index].tat - j1[min_index].bt;

    avg1 += j1[min_index].tat;
    avg2 += j1[min_index].wt;
}
time = time + 1;
}
avg1 = avg1 / n;
avg2 = avg2 / n;
cout<<"ID\tAT\tBT\tST\tCT\tWT\tTAT"<<endl;
for(i=0;i<n;++i)
{
    cout<<j1[i].id<<"\t"<<j1[i].at<<"\t"<<j1[i].bt<<"\t"<<j1[i].st<<"\t"<<j1[i].ct<<"\t"<<j1[i].wt<<"\t"<<j1[i].tat<<endl;
}
}

void non_preemptive(job j1[], float &avg1, float &avg2, int n)
{

```

```

avg1 = 0;
avg2 = 0;
int rem[n],i;
for(i=0;i<n;++i)
{
    rem[i] = j1[i].bt;
}
int completed = 0,time = 0;
while(completed<n)
{
    int min_time = INT_MAX,min_index = -1;
    for(i=0;i<n;++i)
    {
        if(j1[i].at<=time and rem[i]>0 and rem[i]<min_time)
        {
            min_time = rem[i];
            min_index = i;
        }
    }
    if(min_index== -1)
    {
        time = time + 1;
        continue;
    }
    j1[min_index].st = time;
    j1[min_index].ct = time + j1[min_index].bt;
    rem[min_index] = 0;

    time = time + j1[min_index].bt;

    min_time = rem[min_index];

    completed += 1;
    min_time = INT_MAX;

    j1[min_index].tat = j1[min_index].ct - j1[min_index].at;
    j1[min_index].wt = j1[min_index].tat - j1[min_index].bt;

    avg1 += j1[min_index].tat;
    avg2 += j1[min_index].wt;

    //time = time + 1;
}
avg1 = avg1 / n;
avg2 = avg2 / n;
cout<<"ID\tAT\tBT\tST\tCT\tWT\tTAT"<<endl;
for(i=0;i<n;++i)

```

```

    {
        cout<<j1[i].id<<"\t"<<j1[i].at<<"\t"<<j1[i].bt<<"\t"<<j1[i].st<<"\t"<<j1[i].ct<<"\t"<<j1[i]
    ].wt<<"\t"<<j1[i].tat<<endl;
    }
}

int main()
{
    int n,i;
    cout<<"Enter the number of processes: ";
    cin>>n;
    job j1[n];
    cout<<"Enter the id, arrival time and burst time separated by spaces:-"<<endl;
    for(i=0;i<n;++i)
    {
        cin>>j1[i].id>>j1[i].at>>j1[i].bt;
    }
    float avg1,avg2,avg3,avg4;
    cout<<"\nPreemptive job schedule is as follows :-\n";
    preemptive(j1,avg1,avg2,n);
    cout<<"Average turn around time: "<<avg1<<endl;
    cout<<"Average waiting time: "<<avg2<<endl;
    cout<<"\nNo-preemptive job schedule is as follows :-\n";
    non_preemptive(j1,avg3,avg4,n);
    cout<<"Average turn around time: "<<avg3<<endl;
    cout<<"Average waiting time: "<<avg4<<endl;
    cout<<"\nDifference :-\n";
    cout<<"Turn around time: "<<abs(avg1-avg3)<<endl;
    cout<<"Waiting time: "<<abs(avg2-avg4)<<endl;
    return 0;
}

```

## Output

```
vivek@My-PC:~/Desktop$ g++ task.cpp
vivek@My-PC:~/Desktop$ ./a.out
Enter the number of processes: 4
Enter the id, arrival time and burst time separated by spaces:-
1 0 4
2 1 3
3 2 5
4 3 2
Preemptive job schedule is as follows :-
ID    AT      BT      ST      CT      WT      TAT
1     0       4       0       4       0       4
2     1       3       6       9       5       8
3     2       5       9       14      7       12
4     3       2       4       6       1       3
Average turn around time: 6.75
Average waiting time: 3.25
No-preemptive job schedule is as follows :-
ID    AT      BT      ST      CT      WT      TAT
1     0       4       0       4       0       4
2     1       3       6       9       5       8
3     2       5       9       14      7       12
4     3       2       4       6       1       3
Average turn around time: 6.75
Average waiting time: 3.25
Difference :- 0
Turn around time: 0
Waiting time: 0
vivek@My-PC:~/Desktop$
```

```

vivek@My-PC:~/Desktop$ ./a.out
Enter the number of processes: 7
Enter the id, arrival time and burst time separated by spaces:-
1 0 1
2 1 6
3 1 3
4 4 5
5 2 8
6 6 10
7 6 3
Preemptive job schedule is as follows :-
ID AT BT ST CT WT TAT
1 int 0 int 1 0 1 0 1
2 1 6 12 18 11 17
3 1 3 1 4 0 3
4 4 5 cout << "Average turn around time: " << avg1 << endl;
5 2 8 18 26 16 24
6 6 10 26 36 20 30
7 6 3 9 12 3 6
Average turn around time: 12.2857
Average waiting time: 7.14286
No-preemptive job schedule is as follows :-
ID AT BT ST CT WT TAT
1 0 cout << "No-preemptive job schedule is as follows :-" << endl;
2 1 preemptive(1,avg1);
3 1 3 1 4 0 3
4 4 5 Average waiting time: 0 << avg2 << endl;
5 2 8 18 26 16 24
6 6 10 26 36 20 30
7 6 3 Average turn around time: 12.2857 << avg3 << endl;
Average waiting time: 7.14286
Difference :-
Turn around time: 0
Waiting time: 0

```

```

vivek@My-PC:~/Desktop$ ./a.out
Enter the number of processes: 5
Enter the id, arrival time and burst time separated by spaces:-
1 2 6
2 5 2
3 1 8
4 0 3
5 4 4
Preemptive job schedule is as follows :-
ID AT BT ST CT WT TAT
1 2 6 3 15 7 13
2 5 2 5 7 0 2
3 1 8 15 23 14 22
4 0 3 0 3 0 3
5 4 4 4 10 2 6
Average turn around time: 9.2
Average waiting time: 4.6
No-preemptive job schedule is as follows :-
ID AT BT ST CT WT TAT
1 2 6 3 9 1 7
2 5 2 9 11 4 6
3 1 8 15 23 14 22
4 0 3 0 3 0 3
5 4 4 11 15 7 11
Average turn around time: 9.8
Average waiting time: 5.2
Difference :-
Turn around time: 0.6
Waiting time: 0.6
vivek@My-PC:~/Desktop$ 

```

**Result:** Preemptive and non-preemptive shortest job first CPU scheduling algorithms were implemented successfully and verified.

## Experiment Number 7: Implementation of Multilevel Queue Scheduling.

Date:September 22, 2020

Aim:To write the program to implement multilevel queue scheduling taking two queues containing System and User processes respectively where System queue has priority over User queue and each queue is scheduled individually using FCFS.

### C++ Implementation

```
#include<bits/stdc++.h>

using namespace std;

struct process
{
    int at,st,ct,tat,bt,wt,id;
    string type;
};

void find(process p1[], int n)
{
    int time = 0, completed = 0;
    int time1[n],i;
    float avg1 = 0, avg2 = 0;
    bool visited[n] = {};
    for(i=0;i<n;++i)
    {
        time1[i] = p1[i].bt;
    }
    while(completed<n)
    {
        int idx = -1,at = INT_MAX;
        for(i=0;i<n;++i)
        {
            if(p1[i].at <= time and time1[i]>0 and p1[i].type=="System" and p1[i].at <
at)
            {
                idx = i;
                at = p1[i].at;
            }
        }
        if(idx== -1)
        {
            at = INT_MAX;
            for(i=0;i<n;++i)
            {
```

```

        if(p1[i].at <= time and time1[i]>0 and p1[i].type=="User" and
p1[i].at < at)
    {
        idx = i;
        at = p1[i].at;
    }
}
if(idx!=-1)
{
    if(visited[idx]==false)
    {
        p1[idx].st = time;
        visited[idx] = true;
    }

    time = time + 1;
    time1[idx] -= 1;

    if(time1[idx]==0)
    {
        p1[idx].ct = time;

        completed = completed + 1;

        p1[idx].tat = p1[idx].ct - p1[idx].at;
        p1[idx].wt = p1[idx].tat - p1[idx].bt;

        avg1 += p1[idx].tat;
        avg2 += p1[idx].wt;
    }
}
else
{
    time = time + 1;
}
}

cout<<"\nThe task schedule is as follows :-\n";
cout<<"ID\tAT\tBT\tType\tST\tCT\tWT\tTAT"<<endl;
for(i=0;i<n;++i)
{
    cout<<p1[i].id<<"\t"<<p1[i].at<<"\t"<<p1[i].bt<<"\t"<<p1[i].type<<"\t"<<p1[i].st<<"\t"
<<p1[i].ct<<"\t"<<p1[i].wt<<"\t"<<p1[i].tat<<endl;
}
avg1 = avg1;
avg2 = avg2;

```

```

        cout<<"\nAverage turn around time: "<<avg1 / n<<endl;
        cout<<"Average waiting time: "<<avg2 / n<<endl;
    }

int main()
{
    int n,i;
    cout<<"Enter the number of processes: ";
    cin>>n;
    process p1[n];
    cout<<"Enter the id, arrival time and burst time separated by spaces:-"<<endl;
    for(i=0;i<n;++i)
    {
        cin>>p1[i].id>>p1[i].at>>p1[i].bt>>p1[i].type;
    }
    find(p1,n);
    return 0;
}

```

## Output

```

vivek@My-PC:~/Desktop/Operating Systems/Assignment_4$ g++ example_2.cpp
vivek@My-PC:~/Desktop/Operating Systems/Assignment_4$ ./a.out
Enter the number of processes: 6
Enter the id, arrival time and burst time separated by spaces:-
1 0 4 System
2 0 4 User
3 3 3 User
4 4 6 System
5 7 3 System
6 12 5 User

The task schedule is as follows :-
ID      AT      BT      Type     ST      CT      WT      TAT
1       0       4       System   0       4       0       4
2       0       4       User    13      17      13      17
3       3       3       User    17      20      14      17
4       4       6       System   4       10      0       6
5       7       3       System   10      13      3       6
6      12      5       User    20      25      8       13

Average turn around time: 10.5
Average waiting time: 6.33333
vivek@My-PC:~/Desktop/Operating Systems/Assignment_4$ 

```

**Result:** The given multilevel queue scheduling was implemented successfully and verified.

## Experiment Number 8: Implementation of Multilevel Queue Scheduling.

**Date:** September 22, 2020

**Aim:** To write the program to implement multilevel queue scheduling taking two queues containing System and User processes respectively where System queue has priority over User queue and System queue has round robin scheduling whereas User queue has FCFS.

### C++ Implementation

```
#include<bits/stdc++.h>

using namespace std;

struct process
{
    int id,at,bt,st,ct,tat,wt;
    string type;
};

void find(process p1[], int n, int q)
{
    int time = 0, completed = 0, time1[n] = { }, i;
    bool pushed[n] = { }, visited[n] = { };
    for(i=0;i<n;++i)
    {
        time1[i] = p1[i].bt;
    }
    queue <int> q1;
    float avg1 = 0, avg2 = 0;
    for(i=0;i<n;++i)
    {
        if(pushed[i]==false and p1[i].at<=time and p1[i].type=="System")
        {
            q1.push(i);
            pushed[i] = true;
        }
    }
    while(completed<n)
    {
        if(q1.empty()==false)
        {
            int idx = q1.front();
            q1.pop();
            if(visited[idx]==false)
            {
                p1[idx].st = time;
```

```

        visited[idx] = true;
    }
    if(time1[idx]<=q)
    {
        time = time + time1[idx];
        p1[idx].ct = time;
        time1[idx] = 0;

        completed = completed + 1;

        p1[idx].tat = p1[idx].ct - p1[idx].at;
        p1[idx].wt = p1[idx].tat - p1[idx].bt;

        avg1 += p1[idx].tat;
        avg2 += p1[idx].wt;

        for(i=0;i<n;++i)
        {
            if(pushed[i]==false and p1[i].at<=time and
p1[i].type=="System")
            {
                q1.push(i);
                pushed[i] = true;
            }
        }

    }
    else
    {
        time1[idx] = time1[idx] - q;
        time = time + q;
        for(i=0;i<n;++i)
        {
            if(pushed[i]==false and p1[i].at<=time and
p1[i].type=="System")
            {
                q1.push(i);
                pushed[i] = true;
            }
        }
        q1.push(idx);
    }
}
else
{
    int idx = -1, at = INT_MAX;
    for(i=0;i<n;++i)

```

```

{
    if(time1[i]>0 and p1[i].type=="User" and p1[i].at < at)
    {
        idx = i;
        at = p1[i].at;
    }
}
if(idx== -1)
{
    time = time + 1;
    continue;
}
else
{
    if(visited[idx]==false)
    {
        p1[idx].st = time;
        visited[idx] = true;
    }

    time = time + 1;
    time1[idx] -= 1;

    if(time1[idx]==0)
    {
        p1[idx].ct = time;

        completed = completed + 1;

        p1[idx].tat = p1[idx].ct - p1[idx].at;
        p1[idx].wt = p1[idx].tat - p1[idx].bt;

        avg1 += p1[idx].tat;
        avg2 += p1[idx].wt;
    }
}

for(i=0;i<n;++i)
{
    if(pushed[i]==false and p1[i].at<=time and p1[i].type=="System")
    {
        q1.push(i);
        pushed[i] = true;
    }
}
}
}

```

```

cout<<"\nThe task schedule is as follows :-\n";
cout<<"ID\tAT\tBT\tType\tST\tCT\tWT\tTAT"<<endl;
for(i=0;i<n;++i)
{
    cout<<p1[i].id<<"\t"<<p1[i].at<<"\t"<<p1[i].bt<<"\t"<<p1[i].type<<"\t"<<p1[i].st<<"\t"
<<p1[i].ct<<"\t"<<p1[i].wt<<"\t"<<p1[i].tat<<endl;
}
avg1 = avg1;
avg2 = avg2;
cout<<"\nAverage turn around time: "<<avg1 / n<<endl;
cout<<"Average waiting time: "<<avg2 / n<<endl;
}

int main()
{
    int n,i;
    cout<<"Enter the number of processes: ";
    cin>>n;
    process p1[n];
    cout<<"Enter the id, arrival time and burst time separated by spaces:-"<<endl;
    for(i=0;i<n;++i)
    {
        cin>>p1[i].id>>p1[i].at>>p1[i].bt>>p1[i].type;
    }
    find(p1,n,2);
    return 0;
}

```

## Output

```
vivek@My-PC:~/Desktop/Operating Systems/Assignment_4$ g++ example_3.cpp
vivek@My-PC:~/Desktop/Operating Systems/Assignment_4$ ./a.out
Enter the number of processes: 6
Enter the id, arrival time and burst time separated by spaces:-
1 0 6 System
2 2 4 User
3 2 3 System
4 4 5 User
5 7 3 System
6 10 5 User
The task schedule is as follows :-
ID    AT      BT      Type     ST      CT      WT      TAT
1      0       6       System   0       9       3       9
2      2       4       User    12      16      10      14
3      2       3       System   2       7       2       5
4      4       5       User    16      21      12      17
5      7       3       System   9       12      2       5
6      10      5       User    21      26      11      16
Average turn around time: 11
Average waiting time: 6.66667
vivek@My-PC:~/Desktop/Operating Systems/Assignment_4$
```

```

int n,i;
cout<<"Enter the number of processes: ";
cin>>n;
process pl[n];
cout<<"Enter the id, arrival time and burst time separated by spaces: "<<endl;
for(i=0;i<n;++i)
{
    cin>>pl[i].id>>pl[i].at>>pl[i].bt>>pl[i].type;
}
find(pl,n,2);
return 0;
```

**Result:** The given multilevel queue scheduling was implemented successfully and verified.

## **Experiment Number 9: Implementation of Producer & Consumer Problem**

**Date:** September 29, 2020

**Aim:** To implement Producer & Consumer problem (bounded buffer problem) using semaphores.

### **Algorithm for Producer & Consumer problem.**

1	Declare variable for producer & consumer as pthread tid produce and tid consume.
2	Declare a structure to add items, semaphore variable set as struct.
3	Read number the items to be produced and consumed.
4	Declare and define semaphore function for creation and destroy.
5	Define producerfunction.
6	Define consumer function.
7	Call producer andconsumer.
8	Stop theexecution.

### **C++ Implementation**

```
#include<bits/stdc++.h>

using namespace std;

int Mutex=1,full=0,empty=3,x=0;

int main()
{
    int choice;
    void producer_utility();
    void consumer_utility();
    int wait(int),signal(int);
    cout<<"Enter the buffer size: ";
    cin>>empty;
    cout<<"1. Producer"<<endl;
    cout<<"2. Consumer"<<endl;
    cout<<"3. Exit"<<endl;
    while(true)
    {
        cout<<"Enter your choice (1/2/3): ";
        cin>>choice;
        switch(choice)
        {
            case 1:
                if((Mutex==1) and (empty!=0))
                {
```

```

        producer_utility();
    }
else
{
    cout<<"Buffer full!!!"<<endl;
}
break;

case 2:
if((Mutex==1) and (full!=0))
{
    consumer_utility();
}
else
{
    cout<<"Buffer empty!!!"<<endl;
}
break;

case 3:
exit(0);
break;

default:
cout<<"Invalid Choice"<<endl;
}

}

return 0;
}

int wait(int s)
{
    s -= 1;
    return s;
}

int signal(int s)
{
    s += 1;
    return s;
}

void producer_utility()
{
    Mutex = wait(Mutex);
    full = signal(full);
    empty = wait(empty);
}

```

```

        x += 1;
        cout<<"Item "<<x<<" produced!"<<endl;
        Mutex = signal(Mutex);
    }

void consumer_utility()
{
    Mutex = wait(Mutex);
    full = wait(full);
    empty = signal(empty);
    cout<<"Item "<<x<<" consumed!"<<endl;
    x -= 1;
    Mutex = signal(Mutex);
}

```

## Output

```

vivek@My-PC:~/Desktop/Operating Systems/Assignment_5$ ./a.out
Enter the buffer size: 3
1. Producer
2. Consumer
3. Exit
Enter your choice (1/2/3): 1
Item 1 produced!
Enter your choice (1/2/3): 1
Item 2 produced!
Enter your choice (1/2/3): 1
Item 3 produced!
Enter your choice (1/2/3): 1
Buffer full!!!
Enter your choice (1/2/3): 2
Item 3 consumed!
Enter your choice (1/2/3): 2
Item 2 consumed!
Enter your choice (1/2/3): 1
Item 2 produced!
Enter your choice (1/2/3): 2
Item 2 consumed!
Enter your choice (1/2/3): 2
Item 1 consumed!
Enter your choice (1/2/3): 2
Buffer empty!!!
Enter your choice (1/2/3): 3 EXIT
vivek@My-PC:~/Desktop/Operating Systems/Assignment_5$ 

void consumer_utility()
{
    Mutex = wait(Mutex);
    full = wait(full);
    empty = signal(empty);
    cout<<"Item "<<x<<" consumed!"<<endl;
    x -= 1;
    Mutex = signal(Mutex);
}

```

**Result:** The Producer & Consumer problem was implemented and verified successfull

## Experiment Number 10: Implementation of Dining Philosopher's problem using mutex and semaphores.

**Date:** October 6, 2020

**Aim:** To implement Dining Philosopher's problem using mutex and semaphores.

### C++ implementation using mutex.

```
#include <bits/stdc++.h>
#include <mutex>
#include <thread>

using namespace std;
const int N = 8;

class fork
{
public:
    fork()
    {
        ;
    }
    mutex mu;
};

void eat(fork &fork_left, fork& fork_right, int number)
{
    fork_left.mu.lock();
    fork_right.mu.lock();
    cout << "Philosopher " << number << " is eating." << endl;
    this_thread::sleep_for(chrono::seconds(2));
    cout << "\nPhilosopher " << number << " has finished eating." << endl;
    fork_right.mu.unlock();
    fork_left.mu.unlock();
}

int main()
{
    fork forks[N];
    thread philosopher[N];
    cout << "Philosopher " << 1 << " is thinking." << endl;
    philosopher[0] = thread(eat, ref(forks[0]), ref(forks[N-1]), 1);
    for(int i = 1; i < N; ++i)
    {
```

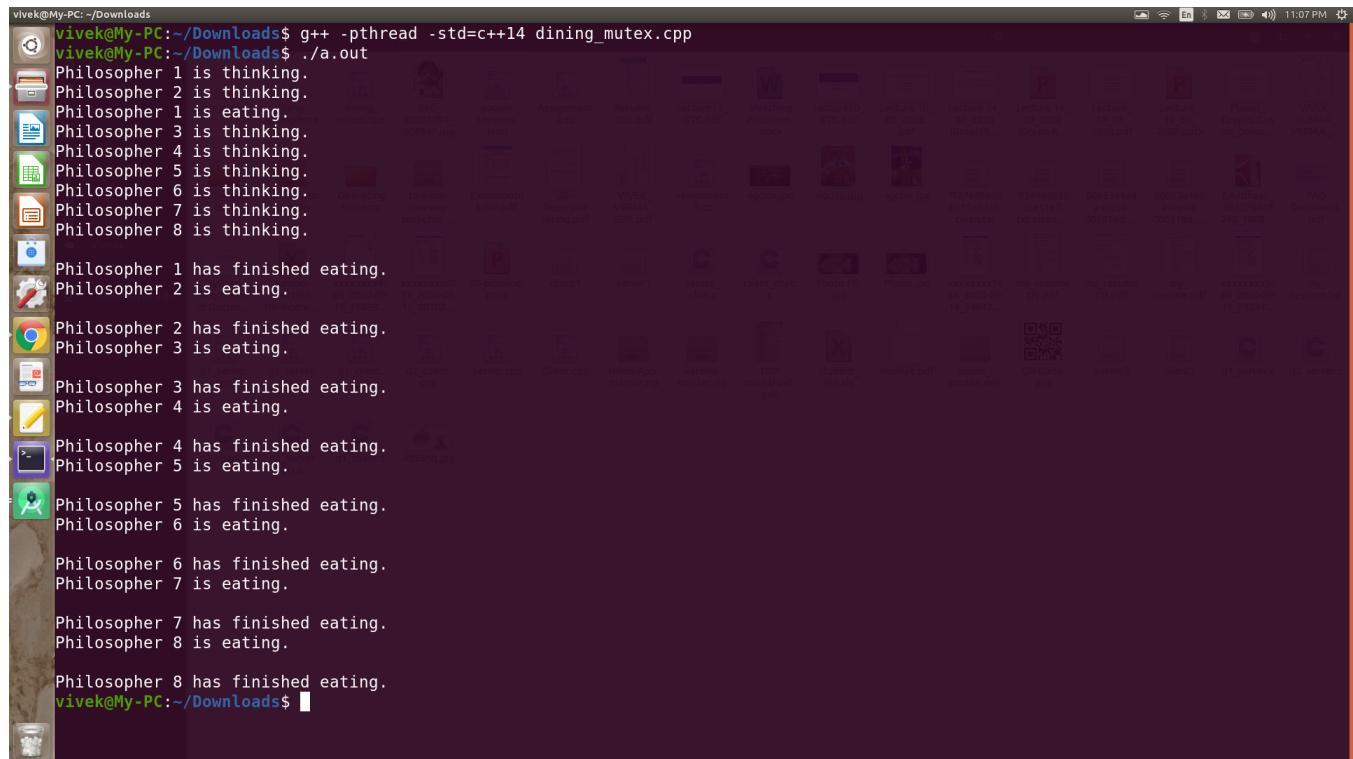
```

        cout << "Philosopher " << (i+1) << " is thinking." << endl;
        philosopher[i] = thread(eat, ref(forks[i]), ref(forks[i-1]), (i+1));
    }

    for(auto &ph: philosopher) ph.join();
    return 0;
}

```

## Output



```

vivek@My-PC:~/Downloads$ g++ -pthread -std=c++14 dining_mutex.cpp
vivek@My-PC:~/Downloads$ ./a.out
Philosopher 1 is thinking.
Philosopher 2 is thinking.
Philosopher 1 is eating.
Philosopher 3 is thinking.
Philosopher 4 is thinking.
Philosopher 5 is thinking.
Philosopher 6 is thinking.
Philosopher 7 is thinking.
Philosopher 8 is thinking.

Philosopher 1 has finished eating.
Philosopher 2 is eating.

Philosopher 2 has finished eating.
Philosopher 3 is eating.

Philosopher 3 has finished eating.
Philosopher 4 is eating.

Philosopher 4 has finished eating.
Philosopher 5 is eating.

Philosopher 5 has finished eating.
Philosopher 6 is eating.

Philosopher 6 has finished eating.
Philosopher 7 is eating.

Philosopher 7 has finished eating.
Philosopher 8 is eating.

Philosopher 8 has finished eating.
vivek@My-PC:~/Downloads$ 

```

## C++ Implementation (using Semaphores)

```

#include <bits/stdc++.h>
#include <mutex>
#include <thread>

using namespace std;
const int N = 8;

class Semaphore
{
private:
    bool signaled;
    pthread_mutex_t m;
    pthread_cond_t c;

```

```

        void Lock()
        {
            pthread_mutex_lock(&m);
        }

        void Unlock()
        {
            pthread_mutex_unlock(&m);
        }

    public:
        Semaphore();
        void P();
        void V();
};

Semaphore::Semaphore()
{
    signaled = true;
    c = PTHREAD_COND_INITIALIZER;
    m = PTHREAD_MUTEX_INITIALIZER;
}

void Semaphore::P()
{
    Lock();
    while (!signaled)
    {
        pthread_cond_wait(&c, &m);
    }
    signaled = false;
    Unlock();
}

void Semaphore::V()
{
    bool previously_signaled;
    Lock();
    previously_signaled = signaled;
    signaled = true;
    Unlock();
    if (!previously_signaled)
        pthread_cond_signal(&c);
}

void eat(Semaphore &fork_left, Semaphore &fork_right, int number)

```

```

{
    fork_left.P();
    fork_right.P();
    cout << "Philosopher " << number << " is eating." << endl;
    this_thread::sleep_for(chrono::seconds(2));
    cout << "\nPhilosopher " << number << " has finished eating." << endl;
    fork_right.V();
    fork_left.V();
}

int main()
{
    Semaphore forks[N];

    thread philosopher[N];

    cout << "Philosopher " << 1 << " is thinking." << endl;
    philosopher[0] = thread(eat, ref(forks[0]), ref(forks[N-1]), 1);

    for(int i = 1; i < N; ++i)
    {
        cout << "Philosopher " << (i+1) << " is thinking." << endl;
        philosopher[i] = thread(eat, ref(forks[i]), ref(forks[i-1]), (i+1));
    }

    for(auto &ph: philosopher) ph.join();
    return 0;
}

```

```
vivek@My-PC:~/Downloads$ g++ -pthread -std=c++14 dining_semaphore.cpp
vivek@My-PC:~/Downloads$ ./a.out
Philosopher 1 is thinking.
Philosopher 2 is thinking.
Philosopher Philosopher 13 is eating. is thinking.

Philosopher 4 is thinking.
Philosopher 5 is thinking.
Philosopher 6 is thinking.
Philosopher 7 is thinking.
Philosopher 8 is thinking.

Philosopher 1 has finished eating.
Philosopher 2 is eating.

Philosopher 2 has finished eating.
Philosopher 3 is eating.

Philosopher 3 has finished eating.
Philosopher 4 is eating.

Philosopher 4 has finished eating.
Philosopher 5 is eating.

Philosopher 5 has finished eating.
Philosopher 6 is eating.

Philosopher 6 has finished eating.
Philosopher 7 is eating.

Philosopher 7 has finished eating.
Philosopher 8 is eating.

Philosopher 8 has finished eating.
vivek@My-PC:~/Downloads$
```

**Result:** The Dining Philosopher's problem was implemented using mutex & semaphores and verified successfully.

## **Experiment Number 11: Implementation of Banker's and Deadlock Prevention Algorithms**

**Date:** October 13, 2020

**Aim:** To implement Banker's and Deadlock prevention algorithm.

### **1. Deadlock Prevention**

- We can prevent Deadlock by eliminating any of the below four conditions.

### **2. Eliminate Mutual Exclusion**

- It is not possible to dis-satisfy the mutual exclusion because some resources, such as the tape drive and printer, are inherently non-shareable.

### **3. Eliminate Hold and wait**

- Allocate all required resources to the process before the start of its execution, this wayhold and wait condition is eliminated but it will lead to low device utilization. For example, if a process requires printer at a later time and we have allocated printer before the start of its execution printer will remain blocked till it has completed its execution.
- The process will make a new request for resources after releasing the current set of resources. This solution may lead to starvation.

### **4. Eliminate No Preemption**

- Preempt resources from the process when resources required by other high priority processes.

### **5. Eliminate Circular Wait**

- Each resource will be assigned with a numerical number. A process can request the resources increasing/decreasing order of numbering.
- For Example, if P1 process is allocated R5 resources, now next time if P1 ask for R4, R3 lesser than R5 such request will not be granted, only request for resources more than R5 will be granted.

### **Algorithm**

1.	Start the program.
2.	Attacking mutex condition: never grant exclusive access. But this may not be possible for several resources.
3.	Attacking preemption: not something you want to do.
4.	Attacking hold and wait condition: make a process hold at the most 1 resource.
5.	At a time. Make all the requests at the beginning. Nothing policy. If you feel, retry.
6.	Attacking circular wait: Order all the resources. Make sure that the requests are issued in the.
7.	Correct order so that there are no cycles present in the resource graph. Resources numbered 1 ... n.
8.	Resources can be requested only in increasing.
9.	Order. i.e. you cannot request a resource whose no is less than any you may be holding.

## C++ Implementation (Baker's algorithm)

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
    int n;
    cout << "Enter the number of processes: ";
    cin >> n;
    int r;
    cout << "Enter the total number of instances present of the resource: ";
    cin >> r;
    vector < vector<int>> safe_sequences;
    vector<int> cur_sequence(n);
    vector<int> allocated(n), maxi(n);
    for(int i = 0; i < n; i++)
    {
        cur_sequence[i] = i;
        cout << "Enter the no. of resources already allocated to P" << i << ": ";
        cin >> allocated[i];
    }

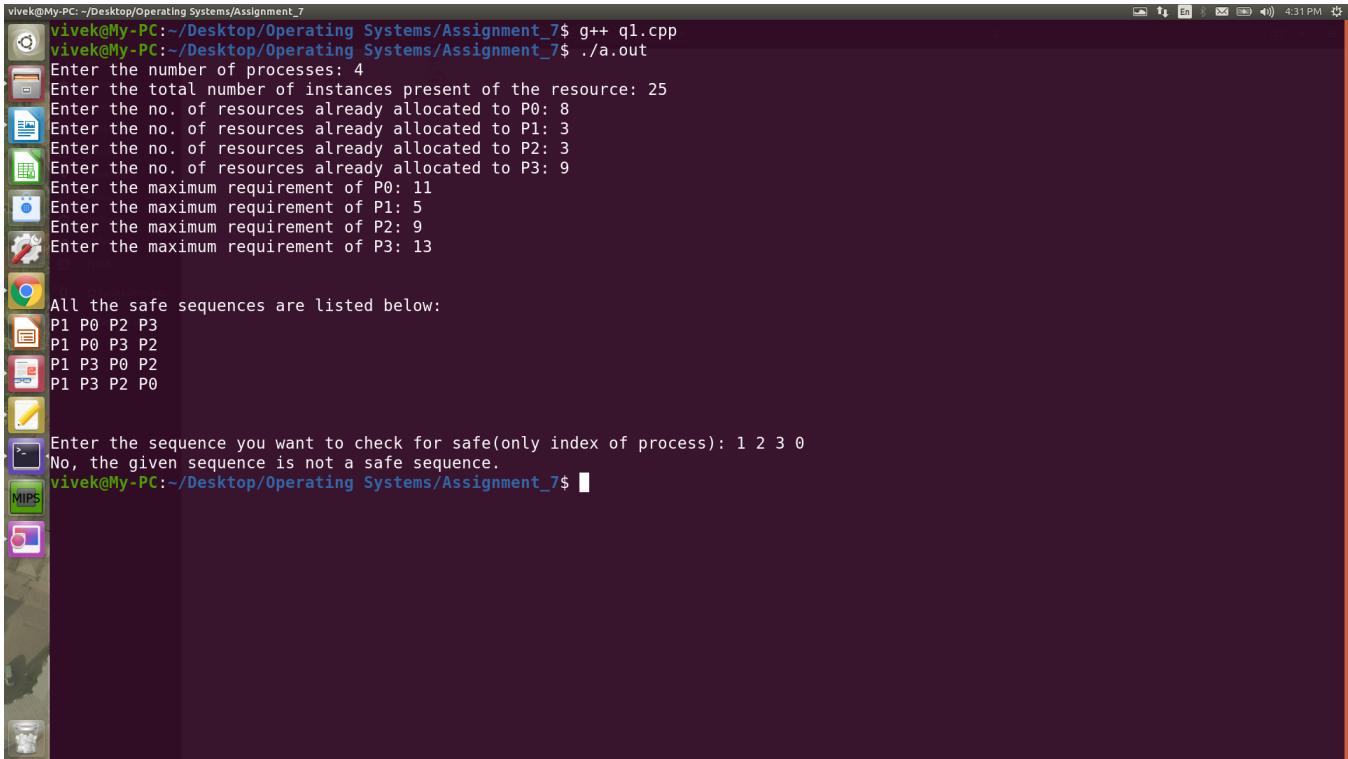
    for(int i = 0; i < n; i++)
    {
        cout << "Enter the maximum requirement of P" << i << ": ";
        cin >> maxi[i];
    }
    while(true)
    {
        int cont = 0; //indicates that current sequence will lead to a deadlock
        vector <int> need(n);
        int available = r;
        for(int i = 0; i < n; i++)
        {
            available -= allocated[i];
            need[i] = maxi[i] - allocated[i];
        }
        for(int i = 0; i < n; i++)
        {
            if(need[cur_sequence[i]] > available)
            {
                cont = 1;
                break;
            }
            available += allocated[cur_sequence[i]];
        }
        if(cont == 1)
            break;
        else
            cout << "Safe Sequence: ";
        for(int i = 0; i < n; i++)
            cout << cur_sequence[i] << " ";
        cout << endl;
        cout << "Enter the next sequence: ";
        cin >> cur_sequence;
    }
}
```

```

        }
        if(cont==0)
        {
            safe_sequences.push_back(cur_sequence);
        }
        int flag = next_permutation(cur_sequence.begin(), cur_sequence.end());
        if(!flag)
        {
            break;
        }
    }
    cout << "\n\nAll the safe sequences are listed below:\n";
    for(int i = 0; i <safe_sequences.size(); i++)
    {
        for(int j = 0; j < n; j++)
        {
            cout <<"P"<<safe_sequences[i][j]<<" ";
        }
        cout<<"\n";
    }
    cout << "\n\nEnter the sequence you want to check for safe(only index of process): ";
    vector<int> check(n);
    for(int i = 0; i < n; i++)
    {
        cin >> check[i];
    }
    int yes = 0;
    for(int i = 0; i < safe_sequences.size(); i++)
    {
        int flag = 0;
        for(int j = 0; j < n; j++)
        {
            if(safe_sequences[i][j] != check[j])
            {
                flag = 1;
                break;
            }
        }
        if(flag == 0)
        {
            yes = 1;
            break;
        }
    }
    if(yes)
    {
        cout <<"Yes, the given sequence is a safe sequence.\n";
    }
}

```

```
    }
else
{
    cout<<"No, the given sequence is not a safe sequence.\n";
}
return 0;
}
```



```
vivek@My-PC:~/Desktop/Operating Systems/Assignment_7$ g++ ql.cpp
vivek@My-PC:~/Desktop/Operating Systems/Assignment_7$ ./a.out
Enter the number of processes: 4
Enter the total number of instances present of the resource: 25
Enter the no. of resources already allocated to P0: 8
Enter the no. of resources already allocated to P1: 3
Enter the no. of resources already allocated to P2: 3
Enter the no. of resources already allocated to P3: 9
Enter the maximum requirement of P0: 11
Enter the maximum requirement of P1: 5
Enter the maximum requirement of P2: 9
Enter the maximum requirement of P3: 13
All the safe sequences are listed below:
P1 P0 P2 P3
P1 P0 P3 P2
P1 P3 P0 P2
P1 P3 P2 P0

Enter the sequence you want to check for safe(only index of process): 1 2 3 0
No, the given sequence is not a safe sequence.
vivek@My-PC:~/Desktop/Operating Systems/Assignment_7$
```

## C++ Implementation of deadlock prevention algorithm

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
    int n;
    cout<<"Enter the number of processes: ";
    cin>>n;
    int r1, r2;
    cout<<"Enter the total instances of resource 1: ";
    cin>>r1;
    cout<<"Enter the total instances of resource 2: ";
    cin>>r2;
    int allocated[n][2], maxi[n][2];
    for(int i = 0; i < n; i++)
    {
        cout<<"Enter the instance(s) of resource 1 already allocated to P" <<i <<": ";
        cin>>allocated[i][0];
        cout<<"Enter the instance(s) of resource 2 already allocated to P" <<i <<": ";
        cin>>allocated[i][1];
    }
}
```

```

for(int i = 0; i < n; i++)
{
    cout<<"Enter the maximum requirement of resource 1 for P"<<i<<": ";
    cin>>maxi[i][0];
    cout<<"Enter the maximum requirement of resource 2 for P"<<i<<": ";
    cin>>maxi[i][1];
}
int available1 = r1, available2 = r2;
for(int i = 0; i < n; i++)
{
    available1 -= allocated[i][0];
    available2 -= allocated[i][1];
}
int need[n][2];
for(int i = 0; i < n; i++)
{
    need[i][0] = maxi[i][0] - allocated[i][0];
    need[i][1] = maxi[i][1] - allocated[i][1];
}
vector<int> safe_sequence;
for(int i = 0; i < n; i++)
{
    int dead_lock = 1;
    for(int j = 0; j < n; j++)
    {
        if(available1 >= need[j][0] && available2 >= need[j][1] && (need[j][0] ||
need[j][1]))
        {
            dead_lock = 0;
            available1 += allocated[j][0];
            need[j][0] = 0;
            available2 += allocated[j][1];
            need[j][1] = 0;
            cout << "\nAfter completion of process P" << j <<", number of
available resource 1 is: "<<available1<<"\n";
            cout << "\nAfter completion of process P" << j <<", number of
available resource 2 is: "<<available2<<"\n";
            safe_sequence.push_back(j);
        }
    }
    if(dead_lock)
    {
        break;
    }
}
if(safe_sequence.size() < n)
{

```

```

        cout << "There is no safe sequence\n";
    }
else
{
    cout<<"The safe sequence is: ";
    for(int i = 0; i < n; i++)
    {
        cout << "P" <<safe_sequence[i]<< " ";
    }
    cout<<"\n";
    cout<<"The last process to be executed is "<<"P"<<safe_sequence[n-1];
}
return 0;
}

```

## Output

```

vivek@My-PC:~/Desktop/Operating Systems/Assignment_7
Enter the number of processes: 4
Enter the total instances of resource 1: 8
Enter the total instances of resource 2: 8
Enter the instance(s) of resource 1 already allocated to P0: 2
Enter the instance(s) of resource 2 already allocated to P0: 1
Enter the instance(s) of resource 1 already allocated to P1: 1
Enter the instance(s) of resource 2 already allocated to P1: 2
Enter the instance(s) of resource 1 already allocated to P2: 0
Enter the instance(s) of resource 2 already allocated to P2: 2
Enter the instance(s) of resource 1 already allocated to P3: 1
Enter the instance(s) of resource 2 already allocated to P3: 0
Enter the maximum requirement of resource 1 for P0: 8
Enter the maximum requirement of resource 2 for P0: 3
Enter the maximum requirement of resource 1 for P1: 3
Enter the maximum requirement of resource 2 for P1: 6
Enter the maximum requirement of resource 1 for P2: 3
Enter the maximum requirement of resource 2 for P2: 4
Enter the maximum requirement of resource 1 for P3: 6
Enter the maximum requirement of resource 2 for P3: 6
After completion of process P2, number of available resource 1 is: 4
After completion of process P2, number of available resource 2 is: 5
After completion of process P1, number of available resource 1 is: 5
After completion of process P1, number of available resource 2 is: 7
After completion of process P3, number of available resource 1 is: 6
After completion of process P3, number of available resource 2 is: 7
After completion of process P0, number of available resource 1 is: 8
After completion of process P0, number of available resource 2 is: 8
The safe sequence is: P2 P1 P3 P0
The last process to be executed is P0
vivek@My-PC:~/Desktop/Operating Systems/Assignment_7$ 

```

**Result:** Banker's and Deadlock Prevention Algorithm was implemented and verified successfully.

## Experiment Number 12: Stimulate Page Replacement Algorithm: FIFO

**Date:** October 20, 2020

**Aim:** To Simulate FIFO page replacement algorithm.

**First in First Out (FIFO):** This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

**Example1:** Consider page reference string 1, 3, 0, 3, 5, 6 with 3-page frames. Find number of page faults.

Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —>**3 Page Faults.**

When 3 comes, it is already in memory so —>**0 Page Faults.**

Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e. 1. —>**1 Page Fault.**

6 comes, it is also not available in memory so it replaces the oldest page slot i.e. 3 —>**1 Page Fault.**

Finally, when 3 come it is not available so it replaces 0 —>**1 Page fault**

**Belady's Anomaly:** Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First out (FIFO) page replacement algorithm.

For example, if we consider reference string 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10-page faults.

### Algorithm

1. Start the program
2. Read the number of frames
3. Read the number of pages
4. Read the page numbers
5. Initialize the values in frames to -1
6. Allocate the pages into frames in First in first out order.
7. Display the number of page faults.
8. Stop the program

## C++ Implementation

```
#include<bits/stdc++.h>

using namespace std;

class Cache
{
private:
    int *cache,n,*time1;
    void increase();

public:
    Cache(int);
    bool find(int x);
    void display();
};

void Cache::display()
{
    cout<<"\t\tThe contents of cache are : ";
    for(int i=0;i<n;++i)
    {
        cout<<cache[i]<<" ";
    }
    cout<<endl;
}

void Cache::increase()
{
    for(int i=0;i<n;++i)
    {
        if(cache[i]!=-1)
        {
            time1[i] = time1[i] + 1;
        }
    }
}

Cache::Cache(int n)
{
    cout<<"\nCache implementing FIFO page replacement policy created!\n";
    this->n = n;
    cache = new int[n];
    time1 = new int[n];
    for(int i=0;i<n;++i)
    {
```

```

        cache[i] = -1;
        time1[i] = INT_MAX;
    }
}

bool Cache::find(int x)
{
    bool hit_miss = false;
    int idx = -1;
    for(i=0;i<n;++i)
    {
        if(cache[i]==x)
        {
            idx = i;
            break;
        }
    }
    if(idx== -1)
    {
        //miss has occurred
        hit_miss = false;
        int last_used = -1, last_used_time = INT_MIN;
        for(i=0;i<n;++i)
        {
            if(time1[i]>last_used_time)
            {
                last_used_time = time1[i];
                last_used = i;
            }
        }
        cache[last_used] = x;
        time1[last_used] = 0;
    }
    else
    {
        //hit has occurred
        hit_miss = true;
        //time1[idx] = 0; the only difference between both algorithms
    }
    this->increase();
    return hit_miss;
}

int main()
{
    int n,f,i,page,faults = 0;
    cout<<"Enter the number of requests: ";

```

```

cin>>n;
cout<<"Enter the frame size: ";
cin>>f;
Cache c1(f);
cout<<"\n";
for(i=0;i<n;++i)
{
    cout<<"Request #"<<i+1<<"\t\t";
    cout<<"Enter the page: ";
    cin>>page;
    if(c1.find(page)==false)
    {
        faults += 1;
        cout<<"      \t\tMiss!"<<endl;
    }
    else
    {
        cout<<"      \t\tHit!"<<endl;
    }
    c1.display();
    cout<<"\n";
}
cout<<"      \t\tPage faults: "<<faults<<endl;
cout<<"      \t\tNumber of hits: "<<n - faults<<endl;
return 0;
}

```

## Output

```
vivek@My-PC:~/Desktop/Operating Systems/Assignment_8$ g++ T1T0.cpp
vivek@My-PC:~/Desktop/Operating Systems/Assignment_8$ ./a.out
Enter the number of requests: 20
Enter the frame size: 3
Cache implementing FIFO page replacement policy created!
Request #1      Enter the page: 1
Miss!
The contents of cache are : 1 -1 -1
Request #2      Enter the page: 2
Miss!
The contents of cache are : 1 2 -1
Request #3      Enter the page: 3
Miss!
The contents of cache are : 1 2 3
Request #4      Enter the page: 2
Hit!
The contents of cache are : 1 2 3
Request #5      Enter the page: 1
Hit!
The contents of cache are : 1 2 3
Request #6      Enter the page: 5
Miss!
The contents of cache are : 5 2 3
Request #7      Enter the page: 2
Hit!
The contents of cache are : 5 2 3
Request #8      Enter the page: 1
Miss!
The contents of cache are : 5 1 3
```

```
vivek@My-PC:~/Desktop/Operating Systems/Assignment_8$ Request #8      Enter the page: 1
Miss!
The contents of cache are : 5 1 3
Request #9      Enter the page: 6
Miss!
The contents of cache are : 5 1 6
Request #10     Enter the page: 2
Miss!
The contents of cache are : 2 1 6
Request #11     Enter the page: 5
Miss!
The contents of cache are : 2 5 6
Request #12     Enter the page: 6
Hit!
The contents of cache are : 2 5 6
Request #13     Enter the page: 3
Miss!
The contents of cache are : 2 5 3
Request #14     Enter the page: 1
Miss!
The contents of cache are : 1 5 3
Request #15     Enter the page: 3
Hit!
The contents of cache are : 1 5 3
Request #16     Enter the page: 6
Miss!
The contents of cache are : 1 6 3
Request #17     Enter the page: 1
Hit!
```

```
vivek@My-PC:~/Desktop/Operating Systems/Assignment_8
The contents of cache are : 2 5 6
Request #13      Enter the page: 3
Miss!
The contents of cache are : 2 5 3
Request #14      Enter the page: 1
Miss!
The contents of cache are : 1 5 3
Request #15      Enter the page: 3
Hit!
The contents of cache are : 1 5 3
Request #16      Enter the page: 6
Miss!
The contents of cache are : 1 6 3
Request #17      Enter the page: 1
Hit!
The contents of cache are : 1 6 3
Request #18      Enter the page: 2
Miss!
The contents of cache are : 1 6 2
Request #19      Enter the page: 4
Miss!
The contents of cache are : 4 6 2
Request #20      Enter the page: 3
Miss!
The contents of cache are : 4 3 2
Page faults: 14
Number of hits: 6
vivek@My-PC:~/Desktop/Operating Systems/Assignment_8$
```

**Result:** FIFO page replacement algorithm was implemented and verified successfully.

## Experiment Number 13: Simulate Page Replacement Algorithm: LRU

**Date:** October 20, 2020

**Aim:** To Simulate LRU page replacement algorithm.

**Least Recently Used:** In this algorithm page will be replaced which is least recently used.

**Example3:** Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4-page frames. Find number of page faults.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —>**4 Page faults**

0 is already there so —>**0 Page fault**.

When 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**

0 is already in memory so —>**0 Page fault**. 4 will takes place of 1 —>**1 Page Fault**

Now for the further page reference string —>**0 Page fault** because they are already available in the memory.

### Algorithm

1. Start
2. Read the number of frames
3. Read the number of pages
4. Read the page numbers
5. Initialize the values in frames to -1
6. Allocate the pages in to frames by selecting the page that has not been used for the longest period of time.
7. Display the number of page faults.
8. Stop

### C++ Implementation

```
#include<bits/stdc++.h>

//implementation differs only on line number 82

using namespace std;

class Cache
{
    private:
```

```

        int *cache,n,*time1;
        void increase();

    public:
        Cache(int);
        bool find(int x);
        void display();
};

void Cache::display()
{
    cout<<"\t\tThe contents of cache are : ";
    for(int i=0;i<n;++i)
    {
        cout<<cache[i]<<" ";
    }
    cout<<endl;
}

void Cache::increase()
{
    for(int i=0;i<n;++i)
    {
        if(cache[i]!=-1)
        {
            time1[i] = time1[i] + 1;
        }
    }
}

Cache::Cache(int n)
{
    cout<<"\nCache implementing LRU page replacement policy created!\n";
    this -> n = n;
    cache = new int[n];
    time1 = new int[n];
    for(int i=0;i<n;++i)
    {
        cache[i] = -1;
        time1[i] = INT_MAX;
    }
}

bool Cache::find(int x)
{
    bool hit_miss = false;
    int idx = -1,i;

```

```

        for(i=0;i<n;++i)
        {
            if(cache[i]==x)
            {
                idx = i;
                break;
            }
        }
        if(idx== -1)
        {
            //miss has occurred
            hit_miss = false;
            int last_used = -1, last_used_time = INT_MIN;
            for(i=0;i<n;++i)
            {
                if(time1[i]>last_used_time)
                {
                    last_used_time = time1[i];
                    last_used = i;
                }
            }
            cache[last_used] = x;
            time1[last_used] = 0;
        }
        else
        {
            //hit has occurred
            hit_miss = true;
            time1[idx] = 0;
        }
        this -> increase();
        return hit_miss;
    }

int main()
{
    int n,f,i,page,faults = 0;
    cout<<"Enter the number of requests: ";
    cin>>n;
    cout<<"Enter the frame size: ";
    cin>>f;
    Cache c1(f);
    cout<<"\n";
    for(i=0;i<n;++i)
    {
        cout<<"Request # "<<i+1<<"\t\t";
        cout<<"Enter the page: ";

```

```

        cin>>page;
        if(c1.find(page)==false)
        {
            faults += 1;
            cout<<"      \t\tMiss!"<<endl;
        }
        else
        {
            cout<<"      \t\tHit!"<<endl;
        }
        c1.display();
        cout<<"\n";
    }
    cout<<"      \tPage faults: "<<faults<<endl;
    cout<<"      \tNumber of hits: "<<n - faults<<endl;
    return 0;
}

```

```

vivek@My-PC:~/Desktop/Operating Systems/Assignment_8$ g++ lru.cpp
vivek@My-PC:~/Desktop/Operating Systems/Assignment_8$ ./a.out
Enter the number of requests: 20
Enter the frame size: 3
Cache implementing LRU page replacement policy created!
Request #1      Enter the page: 1
Miss!
The contents of cache are : 1 -1 -1
Request #2      Enter the page: 2
Miss!
The contents of cache are : 1 2 -1
Request #3      Enter the page: 3
Miss!
The contents of cache are : 1 2 3
Request #4      Enter the page: 2
Hit!
The contents of cache are : 1 2 3
Request #5      Enter the page: 1
Hit!
The contents of cache are : 1 2 3
Request #6      Enter the page: 5
Miss!
The contents of cache are : 1 2 5
Request #7      Enter the page: 2
Hit!
The contents of cache are : 1 2 5
Request #8      Enter the page: 1
Hit!
The contents of cache are : 1 2 5

```

```
vivek@My-PC: ~/Desktop/Operating Systems/Assignment_8
Request #8      Enter the page: 1
Hit!
The contents of cache are : 1 2 5

Request #9      Enter the page: 6
Miss!
The contents of cache are : 1 2 6

Request #10     Enter the page: 2
Hit!
The contents of cache are : 1 2 6

Request #11     Enter the page: 5
Miss!
The contents of cache are : 5 2 6

Request #12     Enter the page: 6
Hit!
The contents of cache are : 5 2 6

Request #13     Enter the page: 3
Miss!
The contents of cache are : 5 3 6

Request #14     Enter the page: 1
Miss!
The contents of cache are : 1 3 6

Request #15     Enter the page: 3
Hit!
The contents of cache are : 1 3 6

Request #16     Enter the page: 6
Hit!
The contents of cache are : 1 3 6

Request #17     Enter the page: 1
Hit!
```

```
vivek@My-PC:~/Desktop/Operating Systems/Assignment_8$ The contents of cache are : 5 2 6
Request #13      Enter the page: 3
                  Miss!
                  The contents of cache are : 5 3 6

Request #14      Enter the page: 1
                  Miss!
                  The contents of cache are : 1 3 6

Request #15      Enter the page: 3
                  Hit!
                  The contents of cache are : 1 3 6

Request #16      Enter the page: 6
                  Hit!
                  The contents of cache are : 1 3 6

Request #17      Enter the page: 1
                  Hit!
                  The contents of cache are : 1 3 6

Request #18      Enter the page: 2
                  Miss!
                  The contents of cache are : 1 2 6

Request #19      Enter the page: 4
                  Miss!
                  The contents of cache are : 1 2 4

Request #20      Enter the page: 3
                  Miss!
                  The contents of cache are : 3 2 4

Page faults: 11
Number of hits: 9
vivek@My-PC:~/Desktop/Operating Systems/Assignment_8$
```

**Result:** LRU Page Replacement Algorithm was implemented and verified successfully.

## Experiment Number 14: Simulate Fixed Partitioning: First Fit, Best Fit and Worst Fit

Date: October 27, 2020

Aim: To simulate first fit, best fit and worst fit on a fixed partitioning disk.

### C++ Implementation

```
#include<bits/stdc++.h>

using namespace std;

class memory_fixed
{
private:
    vector <pair<int,int>> p1;
    vector <int> p2;
    bool *arr;
    int n1, n2;
    vector <int> record;

public:
    memory_fixed(vector<int>&,vector<int>&);
    void first_fit();
    void best_fit();
    void worst_fit();
    void clear();
    void display();
};

void memory_fixed::clear()
{
    int i,j;
    for(i=0;i<n1;++i)
    {
        p1[i].first = 0;
        arr[i] = false;
        record[i] = -1;
    }
}

void memory_fixed::display()
{
    int i,j,internal = 0,external = 0;
    cout<<left<<setw(15)<<"Partition"<<left<<setw(15)<<"Fragment"
```

```

Size" << left << setw(15) << "Capacity" << left << setw(15) << "Free
Space" << left << setw(15) << "Processes" << endl;
    for(i=0;i<n1;++i)
    {
        cout << left << setw(15) << i+1 << left << setw(15) << p1[i].first << left << setw(15) << p1[i].secon
d << left << setw(15) << p1[i].second - p1[i].first;
        if(record[i]==-1)
        {
            cout << "N/A";
        }
        else
        {
            internal += p1[i].second - p1[i].first;
            cout << record[i];
        }
        cout << endl;
        external += p1[i].second - p1[i].first;
    }
    cout << "Total internal fragmentation: " << internal << endl;
    cout << "Total external fragmentation: " << external << endl;
}

memory_fixed::memory_fixed(vector <int>&partitions, vector <int>&processes)
{
    p2 = processes;
    n1 = partitions.size();
    n2 = processes.size();
    arr = new bool[n1];
    int i;
    for(i=0;i<n1;++i)
    {
        p1.push_back({0,partitions[i]});
        arr[i] = false;
        record.push_back(-1);
    }
}

void memory_fixed::first_fit()
{
    clear();
    int i,j,flag = 0;
    for(i=0;i<n2;++i)
    {
        flag = 0;
        for(j=0;j<n1;++j)
        {

```

```

        if(p2[i]<=p1[j].second and arr[j]==false)
        {
            ++flag;
            break;
        }
    }
    if(flag==1)
    {
        arr[j] = true;
        p1[j].first = p2[i];
        record[j] = i + 1;
    }
    else
    {
        cout<<(i+1)<<" "<<" cannot be allocated memory, External
Fragmentation"<<endl;
    }
}

void memory_fixed::best_fit()
{
    clear();
    int i,j,best_size,idx = -1;
    for(i=0;i<n2;++i)
    {
        best_size = INT_MAX;
        idx = -1;
        for(j=0;j<n1;++j)
        {
            if(p1[j].second>= p2[i] and arr[j]==false)
            {
                if(p1[j].second<best_size)
                {
                    best_size = p1[j].second;
                    idx = j;
                }
            }
        }
        if(idx==-1)
        {
            cout<<(i+1)<<" "<<" cannot be allocated memory, External
Fragmentation"<<endl;
        }
        else
        {
            p1[idx].first = p2[i];
        }
    }
}

```

```

        arr[idx] = true;
        record[idx] = i + 1;
    }
}
}

void memory_fixed::worst_fit()
{
    clear();
    int i,j,best_size,idx = -1;
    for(i=0;i<n2;++i)
    {
        best_size = INT_MIN;
        idx = -1;
        for(j=0;j<n1;++j)
        {
            if(p1[j].second>= p2[i] and arr[j]==false)
            {
                if(p1[j].second>best_size)
                {
                    best_size = p1[j].second;
                    idx = j;
                }
            }
        }
        if(idx== -1)
        {
            cout<<(i+1)<<" "<<" cannot be allocated memory, External
Fragmentation"<<endl;
        }
        else
        {
            p1[idx].first = p2[i];
            arr[idx] = true;
            record[idx] = i + 1;
        }
    }
}

int main()
{
    int n1,n2,i;
    cout<<"Enter the number of partitions: ";
    cin>>n1;
    vector <int> v1(n1);
    cout<<"Enter the partition sizes: ";
    for(i=0;i<n1;++i)

```

```
{  
    cin>>v1[i];  
}  
cout<<"Enter the number of processes: ";  
cin>>n2;  
vector <int> v2(n2);  
cout<<"Enter the memory requirement of process: ";  
for(i=0;i<n2;++i)  
{  
    cin>>v2[i];  
}  
memory_fixed m1(v1,v2);  
cout<<"Fixed Partitioning: First Fit"<<endl;  
m1.first_fit();  
m1.display();  
cout<<"\nFixed Partitioning: Best Fit"<<endl;  
m1.best_fit();  
m1.display();  
cout<<"\nFixed Partitioning: Worst Fit"<<endl;  
m1.worst_fit();  
m1.display();  
return 0;  
}
```

## Output

```
vivek@My-PC:~/Desktop/Operating Systems/Assignment_9$ ./a.out
Enter the number of partitions: 5
Enter the partition sizes: 100 500 200 300 600
Enter the number of processes: 4
Enter the memory requirement of process: 212 417 112 405
Fixed Partitioning: First Fit
4 cannot be allocated memory, External Fragmentation
Partition      Fragment Size Capacity    Free Space    Processes
1              100          100          100          N/A
2              500          500          288          1
3              112          200          88           3
4              300          300          188          N/A
5              600          600          183          2
Total internal fragmentation: 559
Total external fragmentation: 959

Fixed Partitioning: Best Fit
Partition      Fragment Size Capacity    Free Space    Processes
1              100          100          100          N/A
2              417          500          83           2
3              112          200          88           3
4              212          300          88           1
5              405          600          195          4
Total internal fragmentation: 454
Total external fragmentation: 554

MIPS Fixed Partitioning: Worst Fit
4 cannot be allocated memory, External Fragmentation
Partition      Fragment Size Capacity    Free Space    Processes
1              100          100          100          N/A
2              417          500          83           2
3              0            200          200          N/A
4              112          300          188          3
5              212          600          388          1
Total internal fragmentation: 659
Total external fragmentation: 959
vivek@My-PC:~/Desktop/Operating Systems/Assignment_9$
```

**Result:** Fixed Partitioning using first fit, best fit and worst fit was implemented and verified successfully.

## Experiment Number 15: Simulate Variable Partitioning: First Fit, Best Fit and Worst Fit

**Date:** October 27, 2020

**Aim:** To simulate first fit, best fit and worst fit on a variable partitioning disk.

### C++ Implementation

```
#include<bits/stdc++.h>

using namespace std;

class memory_variable
{
    private:
        vector <pair<int,int>> p1;
        vector <int> p2;
        bool *arr;
        int n1, n2;
        vector <vector <int>> record;

    public:
        memory_variable(vector<int>&,vector<int>&);
        void first_fit();
        void best_fit();
        void worst_fit();
        void clear();
        void display();
};

void memory_variable::clear()
{
    int i,j;
    for(i=0;i<n1;++i)
    {
        p1[i].first = 0;
        arr[i] = false;
        record[i].clear();
    }
}

void memory_variable::display()
{
    int i,j,internal = 0, external = 0;
    cout<<left<<setw(15)<<"Partition"<<left<<setw(15)<<"Fragment"
```

```

Size" << left << setw(15) << "Capacity" << left << setw(15) << "Free
Space" << left << setw(15) << "Processes" << endl;
    for(i=0;i<n1;++i)
    {
        cout << left << setw(15) << i+1 << left << setw(15) << p1[i].first << left << setw(15) << p1[i].secon
d << left << setw(15) << p1[i].second - p1[i].first;
        for(j=0;j<record[i].size();++j)
        {
            cout << record[i][j] << " ";
        }
        if(record[i].size() == 0)
        {
            cout << "N/A";
        }
        else
        {
            internal += p1[i].second - p1[i].first;
        }
        cout << endl;
        external += p1[i].second - p1[i].first;
    }
    cout << "Total internal fragmentation: " << internal << endl;
    cout << "Total external fragmentation: " << external << endl;
}

memory_variable::memory_variable(vector <int>&partitions, vector <int>&processes)
{
    p2 = processes;
    n1 = partitions.size();
    n2 = processes.size();
    arr = new bool[n1];
    int i;
    for(i=0;i<n1;++i)
    {
        p1.push_back({0,partitions[i]});
        arr[i] = false;
        vector <int> temp;
        record.push_back(temp);
    }
}

void memory_variable::first_fit()
{
    clear();
    int i,j,flag = 0;
    for(i=0;i<n2;++i)
    {

```

```

        flag = 0;
        for(j=0;j<n1;++j)
        {
            if(p2[i]<=p1[j].second - p1[j].first)
            {
                ++flag;
                break;
            }
        }
        if(flag==1)
        {
            p1[j].first += p2[i];
            record[j].push_back(i+1);
        }
        else
        {
            cout<<(i+1)<<" "<<" cannot be allocated memory, External
Fragmentation"<<endl;
        }
    }
}

void memory_variable::best_fit()
{
    clear();
    int i,j,best_size,idx = -1;
    for(i=0;i<n2;++i)
    {
        best_size = INT_MAX;
        idx = -1;
        for(j=0;j<n1;++j)
        {
            if(p1[j].second - p1[j].first>= p2[i])
            {
                if(p1[j].second<best_size)
                {
                    best_size = p1[j].second - p1[j].first;
                    idx = j;
                }
            }
        }
        if(idx==-1)
        {
            cout<<(i+1)<<" "<<" cannot be allocated memory, External
Fragmentation"<<endl;
        }
        else
    }
}

```

```

        {
            p1[idx].first += p2[i];
            record[idx].push_back(i+1);
        }
    }

void memory_variable::worst_fit()
{
    clear();
    int i,j,best_size,idx = -1;
    for(i=0;i<n2;++i)
    {
        best_size = INT_MIN;
        idx = -1;
        for(j=0;j<n1;++j)
        {
            if(p1[j].second - p1[j].first >= p2[i])
            {
                if(p1[j].second - p1[j].second >= best_size)
                {
                    best_size = p1[j].second - p1[j].second;
                    idx = j;
                }
            }
        }
        if(idx == -1)
        {
            cout<<(i+1)<<" "<<" cannot be allocated memory, External
Fragmentation"<<endl;
        }
        else
        {
            p1[idx].first += p2[i];
            record[idx].push_back(i + 1);
        }
    }
}

int main()
{
    int n1,n2,i;
    cout<<"Enter the number of partitions: ";
    cin>>n1;
    vector <int> v1(n1);
    cout<<"Enter the partition sizes: ";
    for(i=0;i<n1;++i)

```

```
{  
    cin>>v1[i];  
}  
cout<<"Enter the number of processes: ";  
cin>>n2;  
vector <int> v2(n2);  
cout<<"Enter the memory requirement of process: ";  
for(i=0;i<n2;++i)  
{  
    cin>>v2[i];  
}  
memory_variable m1(v1,v2);  
cout<<"Variable Partitioning: First Fit"<<endl;  
m1.first_fit();  
m1.display();  
cout<<"\nVariable Partitioning: Best Fit"<<endl;  
m1.best_fit();  
m1.display();  
cout<<"\nVariable Partitioning: Worst Fit"<<endl;  
m1.worst_fit();  
m1.display();  
return 0;  
}
```

## Output

```
vivek@My-PC:~/Desktop/Operating Systems/Assignment_9$ ./a.out
Enter the number of partitions: 5
Enter the partition sizes: 100 500 200 300 600
Enter the number of processes: 4
Enter the memory requirement of process: 212 417 112 405
Variable Partitioning: First Fit
4 cannot be allocated memory, External Fragmentation
Partition      Fragment Size Capacity   Free Space   Processes
1             cout<int> v1(1)    100        100       N/A
2             cout<int> v2(2)    500        176       1 3
3             cout<int> v3(3)    200        200       N/A
4             cout<int> v4(4)    300        300       N/A
5             cout<int> v5(5)    600        183       2
Total internal fragmentation: 359
Total external fragmentation: 959
Variable Partitioning: Best Fit
Partition      Fragment Size Capacity   Free Space   Processes
1             cout<int> v1(1)    100        100       N/A
2             cout<int> v2(2)    500        83        2
3             cout<int> v3(3)    200        88        3
4             cout<int> v4(4)    300        88        1
5             cout<int> v5(5)    600        195       4
Total internal fragmentation: 454
Total external fragmentation: 554
MIPS Variable Partitioning: Worst Fit
4 cannot be allocated memory, External Fragmentation
Partition      Fragment Size Capacity   Free Space   Processes
1             cout<int> v1(1)    100        100       N/A
2             cout<int> v2(2)    500        83        2
3             cout<int> v3(3)    200        200       N/A
4             cout<int> v4(4)    300        300       N/A
5             cout<int> v5(5)    600        276       1 3
Total internal fragmentation: 359
Total external fragmentation: 959
vivek@My-PC:~/Desktop/Operating Systems/Assignment_9$
```

**Result:** Variable Partitioning using first fit, best fit and worst fit was implemented and verified successfully.

