**2018UCO1665**

**Rahul Agarwal**

**COE-3**

**Sem 4**

**OS tutorial file**

**CEC14**

**INDEX**

## FORK

```
#include<iostream>
#include<sys/types.h>
#include<unistd.h>
using namespace std;

int main()
{
   int check=fork();
   if(check==0)//child
   {
      cout<<"I am child process";
   }
   else if(check>0)
   {
      cout<<"\nI am the parent process";
   }
   else
   {
      cout<<"Child cannot be created";
   }
   cout<<"\ndone";
}
```

```
RAHUL@DESKTOP-J4I9US8 /cygdrive/c/sem 4 os
$ ./process_fork

I am child process
I am the parent process
donedone
```

# WAIT PROCESS

```
#include<iostream>
#include<sys/wait.h>
#include<unistd.h>
using namespace std;
int main()
{
   cout<<"Parent process creating child process\n";
   if(fork()>0)
   {
      cout<<"I am parent process and will wait till my child completes the counting\n";
      pid_t id= wait(NULL);
      cout<<"exiting parent after child whose id is:"<<id<<endl;
   }
   else
   {
      cout<<"I am child process and will count from 1 till 10 and then exit\n";
      for(int i=1;i<=10;i++)
      {
         cout<<i<<" ";
      }
      cout<<"\nexiting child with id:"<<getpid()<<endl;
   }
}
```

```
RAHUL@DESKTOP-J4I9US8 /cygdrive/c/sem 4 os
$ ./waitprocess
Parent process creating child process
I am parent process and will wait till my child completes the counting
I am child process and will count from 1 till 10 and then exit
1 2 3 4 5 6 7 8 9 10
exiting child with id:1432
exiting parent after child whose id is:1432
```

# STATUS

```
#include<iostream>
#include<sys/wait.h>
#include<sys/types.h>
#include<unistd.h>
using namespace std;
int main()
{
   if(fork()==0)
   {
      cout<<"Child process id:"<<getpid()<<" performing some commands"<<endl;
      //execl("/bin/sh", "bin/sh", "-c", "./wrongpath", "NULL");
//////comment this line to see the status when child process works fine
   }
   else
   {
      int child_id;
      int status;
      child_id=wait(&status);
      if(WIFEXITED(status)==true)
      {cout<<"child process id:"<<child_id<<" terminated with status:"<<WEXITSTATUS(status);
         if(WEXITSTATUS(status)==0)
         {cout<<" =>Child process commands successful";}
         if(WEXITSTATUS(status)==127)
         {cout<<" =>Invalid path";}}}}
```

```
RAHUL@DESKTOP-J4I9US8 /cygdrive/c/sem 4 os
$ ./status
Child process id:1435 performing some commands
NULL: ./wrongpath: No such file or directory
child process id:1435 terminated with status:127 =>Invalid path
```

## PRINT ID

```
#include<iostream>
#include<sys/types.h>
#include<unistd.h>
using namespace std;

int main()
{
   cout<<"Process id:"<<getpid()<<endl;
   cout<<"Creating child process"<<endl;
   if(fork()==0)
   {
      int p_pid=getppid();
      cout<<"After creating child process"<<endl;
      cout<<"Parent process id:"<<p_pid<<endl;
      cout<<"Child process created id:"<<getpid()<<endl;

   }
}
```

```
RAHUL@DESKTOP-J4I9US8 /cygdrive/c/sem 4 os
$ ./printid
Process id:1437
Creating child process
After creating child process
Parent process id:1437
Child process created id:1438
```

# FIRST COME FIRST SERVE

```
#include<iostream>
using namespace std;
void findWaitingTime(int processes[], int n,
int bt[], int wt[])
{wt[0] = 0;
	for (int i = 1; i < n ; i++ ) wt[i] = bt[i-1] + wt[i-1] ;}
void findTurnAroundTime( int processes[], int n,int bt[], int wt[], int tat[])
{for (int i = 0; i < n ; i++)tat[i] = bt[i] + wt[i];}
void solution( int processes[], int n, int bt[])
{
	int wt[n], tat[n], total_wt = 0, total_tat = 0;
	findWaitingTime(processes, n, bt, wt);
	findTurnAroundTime(processes, n, bt, wt, tat);
	cout << "Processes "<< " Burst time "
		<< " Waiting time " << " Turn around time\n";
	for (int i=0; i<n; i++)
	{
		total_wt = total_wt + wt[i];
		total_tat = total_tat + tat[i];
		cout << " " << i+1 << "\t\t" << bt[i] <<"\t "<< wt[i] <<"\t\t " << tat[i] <<endl;
	}
	cout << "Average waiting time = "<< (float)total_wt / (float)n;
	cout << "\nAverage turn around time = "
		<< (float)total_tat / (float)n;
}
int main()
{   int n;
    cout <<"enter the number of processes"<<endl;
    cin>>n;
   int processes[100];
  cout<<"enter the processes"<<endl;
for(int i = 0 ; i<n ; i++){cin>>processes[i];}
```

```
 cout<<"enter the bursts times"<<endl;

int burst_time[100] ;

for(int i = 0 ; i<n ; i++){cin>>burst_time[i];}

solution(processes, n, burst_time);

return 0;

}
```

```
enter the number of processes
3
enter the processes
1
2
3
enter the bursts times
24
3
3
Processes  Burst time  Waiting time  Turn around time
 1              24       0               24
 2              3        24              27
 3              3        27              30
Average waiting time = 17
Average turn around time = 27
Process returned 0 (0x0)   execution time : 92.364 s
Press any key to continue.
```

## NON-PREEMPTIVE-SJF

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

using namespace std;


class process

{

public:

   int burst_time,arrival_time,index;

};


bool cmp(process a,process b)

{

   if(a.arrival_time==b.arrival_time)
```

```
    {return a.burst_time<b.burst_time; }
    else
    {return a.arrival_time<b.arrival_time; }
}
int main()
{
    cout<<"Enter the number of processes:";
    int n;cin>>n;
    process arr[n];
    cout<<"Enter the arrival time and burst time\n";
    for(int i=0;i<n;i++)
    {cin>>arr[i].arrival_time>>arr[i].burst_time; arr[i].index=i; }
    sort(arr,arr+n,cmp);
    int ans[n];
    memset(ans,0,sizeof(int)*n);
    int time=0;
    for(int i=0;i<n;i++)
    {ans[i]=time-arr[i].arrival_time; time+=arr[i].burst_time; }
    int avg=0;
    for(int i=0;i<n;i++)
    {cout<<"Waiting time of process "<<i+1<<" is "<<ans[i]<<endl; avg+=ans[i]; }
    cout<<"Average waiting time is "<<avg/n;}
```

```
Enter the number of processes:4
Enter the arrival time and burst time
0 6
0 8
0 7
0 3
Waiting time of process 1 is 0
Waiting time of process 2 is 3
Waiting time of process 3 is 9
Waiting time of process 4 is 16
Average waiting time is 7
Process returned 0 (0x0)   execution time : 266.162 s
Press any key to continue.
```

# PREEMPTIVE-SJF

```
#include <bits/stdc++.h>
using namespace std;
struct Process {
	int pid; // Process ID
	int bt; // Burst Time
	int art; // Arrival Time
};
void findWaitingTime(Process proc[], int n,int wt[])
{
	int rt[n];
	for (int i = 0; i < n; i++)
		rt[i] = proc[i].bt;
	int complete = 0, t = 0, minm = INT_MAX;
	int shortest = 0, finish_time;
	bool check = false;
	while (complete != n) {
		for (int j = 0; j < n; j++) {
			if ((proc[j].art <= t) &&(rt[j] < minm) && rt[j] > 0) {
				minm = rt[j];shortest = j;check = true;}}
		if (check == false) {
			t++;
			continue;
		}
		rt[shortest]--;
		minm = rt[shortest];
		if (minm == 0)minm = INT_MAX;
		if (rt[shortest] == 0) {
			complete++;
			check = false;
			finish_time = t + 1;
			wt[shortest] = finish_time -proc[shortest].bt -proc[shortest].art;
			if (wt[shortest] < 0){wt[shortest] = 0;}
		t++;}}
```

```
void findTurnAroundTime(Process proc[], int n,
int wt[], int tat[])
{
        for (int i = 0; i < n; i++)tat[i] = proc[i].bt + wt[i];
}
void findavgTime(Process proc[], int n)
{
        int wt[n], tat[n], total_wt = 0,
        total_tat = 0;
        findWaitingTime(proc, n, wt);
        findTurnAroundTime(proc, n, wt, tat);
        cout << "Processes "<< " Burst time "<< " Waiting time "<< " Turn around time\n";
        for (int i = 0; i < n; i++) {
                total_wt = total_wt + wt[i];
                total_tat = total_tat + tat[i];
                cout << " " << proc[i].pid << "\t\t"<< proc[i].bt << "\t\t " << wt[i]<< "\t\t " << tat[i] << endl;
        }
        cout << "\nAverage waiting time = "<< (float)total_wt / (float)n;
        cout << "\nAverage turn around time = "<< (float)total_tat / (float)n;
}
int main()
{   int n;
   cout <<"enter the number of processes"<<endl;
   cin>>n;
   Process p [10];
   cout<<"enter processes details"<<endl;
   for(int i = 0 ; i< n ; i++)
      {
      cout<<"enter process id"<<endl;
      cin>>p[i].pid;

      cout<<"enter the burst time"<<endl;
      cin>>p[i].bt;
```

```
        cout<<"enter the arrival time"<<endl;
        cin>>p[i].art;
    }

            findavgTime(p, n);
            return 0;
}
```

```
"C:\sem 4 os\sjf_pre-emptive.exe"
0
enter process id
2
enter the burst time
4
enter the arrival time
1
enter process id
3
enter the burst time
9
enter the arrival time
2
enter process id
4
enter the burst time
5
enter the arrival time
3
Processes  Burst time  Waiting time  Turn around time
 1              8              9              17
 2              4              0              4
 3              9              15             24
 4              5              2              7

Average waiting time = 6.5
Average turn around time = 13
Process returned 0 (0x0)   execution time : 53.278 s
```

## PRIORITY FCFS

```
#include<bits/stdc++.h>
using namespace std;
struct Process
{
            int pid; // Process ID
            int bt; // CPU Burst time required
            int priority; // Priority of this process
```

```
};

bool comparison(Process a, Process b)
{return (a.priority < b.priority);}

void findWaitingTime(Process proc[], int n,int wt[])
{

	wt[0] = 0;
	for (int i = 1; i < n ; i++ )wt[i] = proc[i-1].bt + wt[i-1] ;
}
void findTurnAroundTime( Process proc[], int n,int wt[], int tat[])
{for (int i = 0; i < n ; i++)tat[i] = proc[i].bt + wt[i];}

void findavgTime(Process proc[], int n)
{
	int wt[n], tat[n], total_wt = 0, total_tat = 0;
	findWaitingTime(proc, n, wt);
	findTurnAroundTime(proc, n, wt, tat);
	cout << "\nProcesses "<< " Burst time "<< " Waiting time " << " Turn around time\n";
	for (int i=0; i<n; i++)
	{
		total_wt = total_wt + wt[i];
		total_tat = total_tat + tat[i];
		cout << " " << proc[i].pid << "\t\t"
			<< proc[i].bt << "\t " << wt[i]
			<< "\t\t " << tat[i] <<endl;
	}
	cout << "\nAverage waiting time = "<< (float)total_wt / (float)n;
	cout << "\nAverage turn around time = "<< (float)total_tat / (float)n;
}
void priorityScheduling(Process proc[], int n)
{
	sort(proc, proc + n, comparison);
```

```
        cout<< "Order in which processes gets executed \n";
        for (int i = 0 ; i < n; i++)cout << proc[i].pid <<" " ;
        findavgTime(proc, n);
}
int main()
{
        int n;
  cout <<"enter the number of processes"<<endl;
  cin>>n;
  Process p [10];
  cout<<"enter processes details"<<endl;
  for(int i = 0 ; i< n ; i++)
  {
    cout<<"enter process id:";
    cin>>p[i].pid;
    cout<<"enter the burst time:";
    cin>>p[i].bt;
    cout<<"enter the priority:";
    cin>>p[i].priority;}
        priorityScheduling(p, n);
        return 0;}
```

```
"C:\sem 4 os\priority.exe"
enter processes details
enter process id:1
enter the burst time:10
enter the priority:3
enter process id:2
enter the burst time:1
enter the priority:1
enter process id:3
enter the burst time:2
enter the priority:4
enter process id:4
enter the burst time:1
enter the priority:5
enter process id:5
enter the burst time:5
enter the priority:2
Order in which processes gets executed
2 5 1 3 4
Processes  Burst time  Waiting time  Turn around time
 2              1        0              1
 5              5        1              6
 1              10       6              16
 3              2        16             18
 4              1        18             19

Average waiting time = 8.2
Average turn around time = 12
```

# ROUND ROBIN

```
#include<iostream>
using namespace std;
void findWaitingTime(int processes[], int n,int bt[], int wt[], int quantum)
{
   int rem_bt[n];
   for (int i = 0 ; i < n ; i++)rem_bt[i] =  bt[i];
   int t = 0;
   while (1)
   {
      bool done = true;
      for (int i = 0 ; i < n; i++)
      {
         if (rem_bt[i] > 0)
         {
            done = false;
            if (rem_bt[i] > quantum)
            {
               t += quantum;
               rem_bt[i] -= quantum;
            }
            else
            {
               t = t + rem_bt[i];
               wt[i] = t - bt[i];
               rem_bt[i] = 0;
            }
         }
      }
      if (done == true)break;
   }
}
void findTurnAroundTime(int processes[], int n,int bt[], int wt[], int tat[])
{for (int i = 0; i < n ; i++)tat[i] = bt[i] + wt[i];}
```

```
void findavgTime(int processes[], int n, int bt[],int quantum)
{
   int wt[n], tat[n], total_wt = 0, total_tat = 0;
   findWaitingTime(processes, n, bt, wt, quantum);
   findTurnAroundTime(processes, n, bt, wt, tat);
  cout << "Processes "<< " Burst time "<< " Waiting time " << " Turn around time\n";
   for (int i=0; i<n; i++)
   {
      total_wt = total_wt + wt[i];
      total_tat = total_tat + tat[i];
      cout << " " << i+1 << "\t\t" << bt[i] <<"\t "<< wt[i] <<"\t\t " << tat[i] <<endl;
   }
   cout << "Average waiting time = "<< (float)total_wt / (float)n;
   cout << "\nAverage turn around time = "<< (float)total_tat / (float)n;
}
int main()
{
   int processes[] = { 1, 2, 3};
   int n = sizeof processes / sizeof processes[0];
   int burst_time[] = {24,3,3};
   int quantum = 4;
   findavgTime(processes, n, burst_time, quantum);
   return 0;
}
```

```
"C:\sem 4 os\round_robin.exe"
Processes  Burst time  Waiting time  Turn around time
 1              24        6               30
 2              3         4               7
 3              3         7               10
Average waiting time = 5.66667
Average turn around time = 15.6667
```

# READER WRITER PROBLEM

```
#include<stdio.h>
#include<pthread.h>
#include<semaphore.h>
sem_t mutex,writeblock;
int data = 0,rcount = 0;
void *reader(void *arg)
{
  int f;
  f = ((int)arg);
  sem_wait(&mutex);
  rcount = rcount + 1;
  if(rcount==1)
   sem_wait(&writeblock);
  sem_post(&mutex);
  printf("Data read by the reader%d is %d\n",f,data);
  sem_wait(&mutex);
  rcount = rcount - 1;
  if(rcount==0)
   sem_post(&writeblock);
  sem_post(&mutex);
}
void *writer(void *arg)
{
  int f;
  f = ((int) arg);
  sem_wait(&writeblock);
  data++;
  printf("Data writen by the writer%d is %d\n",f,data);
  sem_post(&writeblock);
}
int main()
{
  int i,b;
```

```
 pthread_t rtid[5],wtid[5];
 sem_init(&mutex,0,1);
 sem_init(&writeblock,0,1);
 for(i=0;i<=2;i++)
 {pthread_create(&wtid[i],NULL,writer,(void *)i);pthread_create(&rtid[i],NULL,reader,(void *)i);}
 for(i=0;i<=2;i++)
 {pthread_join(wtid[i],NULL);pthread_join(rtid[i],NULL);}
 return 0;
}
```

```
"C:\sem 4 os\readers_writers.exe"
Data writen by the writer0 is 1
Data read by the reader0 is 1
Data read by the reader1 is 1
Data read by the reader2 is 1
Data writen by the writer1 is 2
Data writen by the writer2 is 3
```

## DINING PHILOSOPHER PROBLEM

```
#include<stdio.h>
#define n 4
int compltedPhilo = 0,i;
struct fork{
        int taken;
}ForkAvil[n];
struct philosp{
        int left;
        int right;
}Philostatus[n];
void goForDinner(int philID){
        if(Philostatus[philID].left==10 && Philostatus[philID].right==10)
    printf("Philosopher %d completed his dinner\n",philID+1);
        else if(Philostatus[philID].left==1 && Philostatus[philID].right==1){
      printf("Philosopher %d completed his dinner\n",philID+1);
      Philostatus[philID].left = Philostatus[philID].right = 10;
      int otherFork = philID-1;
```

```
    if(otherFork== -1)
        otherFork=(n-1);
    ForkAvil[philID].taken = ForkAvil[otherFork].taken = 0;
    printf("Philosopher %d released fork %d and fork %d\n",philID+1,philID+1,otherFork+1);
    compltedPhilo++;
}
else if(Philostatus[philID].left==1 && Philostatus[philID].right==0){
        if(philID==(n-1)){
            if(ForkAvil[philID].taken==0){
                ForkAvil[philID].taken = Philostatus[philID].right = 1;
                printf("Fork %d taken by philosopher %d\n",philID+1,philID+1);
            }else{
                printf("Philosopher %d is waiting for fork %d\n",philID+1,philID+1);
            }
        }else{
            int dupphilID = philID;
            philID-=1;
            if(philID== -1)
                philID=(n-1);
            if(ForkAvil[philID].taken == 0){
                ForkAvil[philID].taken = Philostatus[dupphilID].right = 1;
                printf("Fork %d taken by Philosopher %d\n",philID+1,dupphilID+1);
            }else{
                printf("Philosopher %d is waiting for Fork %d\n",dupphilID+1,philID+1);
            }
        }
    }
    else if(Philostatus[philID].left==0){
            if(philID==(n-1)){
                if(ForkAvil[philID-1].taken==0){
                    ForkAvil[philID-1].taken = Philostatus[philID].left = 1;
                    printf("Fork %d taken by philosopher %d\n",philID,philID+1);
                }else{
                    printf("Philosopher %d is waiting for fork %d\n",philID+1,philID);
```

```
                }
            }else{
                if(ForkAvil[philID].taken == 0){
                    ForkAvil[philID].taken = Philostatus[philID].left = 1;
                    printf("Fork %d taken by Philosopher %d\n",philID+1,philID+1);
                }else{printf("Philosopher %d is waiting for Fork %d\n",philID+1,philID+1); }}}
}
int main(){
        for(i=0;i<n;i++)
    ForkAvil[i].taken=Philostatus[i].left=Philostatus[i].right=0;
        while(compltedPhilo<n){ for(i=0;i<n;i++)goForDinner(i);
        printf("\nTill now num of philosophers completed dinner are %d\n\n",compltedPhilo);}
        return 0;
}
```

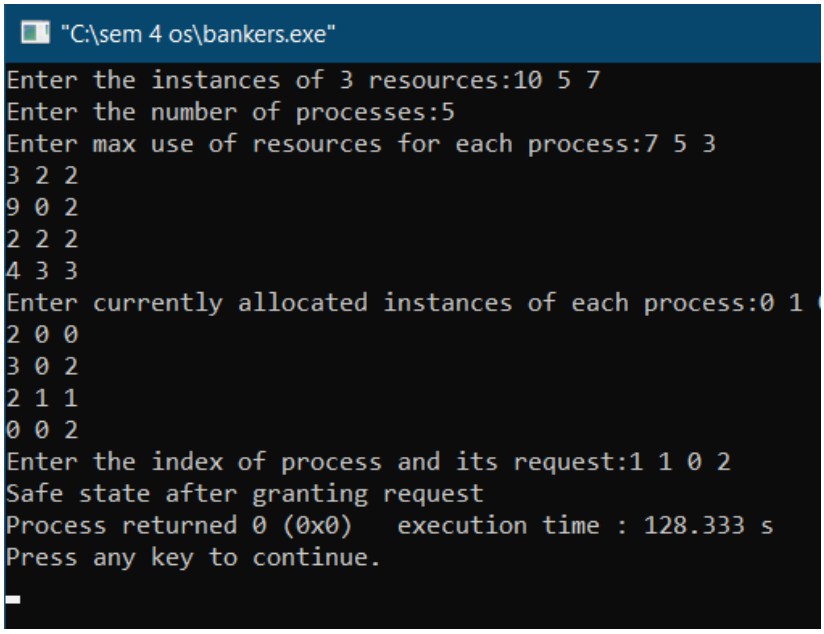
```
"C:\sem 4 os\dining_philosophers.exe"
Fork 1 taken by Philosopher 1
Fork 2 taken by Philosopher 2
Fork 3 taken by Philosopher 3
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Fork 4 taken by Philosopher 1
Philosopher 2 is waiting for Fork 1
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 0

Philosopher 1 completed his dinner
Philosopher 1 released fork 1 and fork 4
Fork 1 taken by Philosopher 2
Philosopher 3 is waiting for Fork 2
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 1

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 2 released fork 2 and fork 1
Fork 2 taken by Philosopher 3
Philosopher 4 is waiting for fork 3

Till now num of philosophers completed dinner are 2

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 3 released fork 3 and fork 2
Fork 3 taken by philosopher 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Fork 4 taken by philosopher 4

Till now num of philosophers completed dinner are 3

Philosopher 1 completed his dinner
Philosopher 2 completed his dinner
Philosopher 3 completed his dinner
Philosopher 4 completed his dinner
Philosopher 4 released fork 4 and fork 3
```

# BANKER ALGORITHM

```
#include<iostream>
using namespace std;
bool banker(int available[3],int allocate[][3],int need[][3],int n)
{
   bool finish[n];
   for(int z=0;z<n;z++){finish[z]=false;}
   int i=0;
   bool again=true;
   while(again==true)
   {
      again=false;
      for(int s=0;s<n;s++)
      {
         if(finish[i]==false && need[i][0]<=available[0] && need[i][1]<=available[1] && need[i][2]<=available[2])
         {
            again=true;
            finish[i]=true;
            available[0]+=allocate[i][0];available[1]+=allocate[i][1];available[2]+=allocate[i][2];
            i=(i+1)%n;
            break;
         }
         i=(i+1)%n;
      }
   }
   for(int z=0;z<n;z++)
   {if(finish[z]==false){return false;} }
   return true;
}
int main()
{
   cout<<"Enter the instances of 3 resources:";
   int available[3];
   cin>>available[0]>>available[1]>>available[2];
```

```
cout<<"Enter the number of processes:";
int n;cin>>n;
cout<<"Enter max use of resources for each process:";
int maxr[n][3];
for(int i=0;i<n;i++)
{cin>>maxr[i][0]>>maxr[i][1]>>maxr[i][2]; }
cout<<"Enter currently allocated instances of each process:";
int allocate[n][3];
for(int i=0;i<n;i++)
{
   cin>>allocate[i][0]>>allocate[i][1]>>allocate[i][2];
   available[0]-=allocate[i][0];
   available[1]-=allocate[i][1];
   available[2]-=allocate[i][2];
}
int need[n][3];
for(int i=0;i<n;i++)
{
   need[i][0]=maxr[i][0]-allocate[i][0];
   need[i][1]=maxr[i][1]-allocate[i][1];
   need[i][2]=maxr[i][2]-allocate[i][2];
}
bool safe=banker(available,allocate,need,n);
if(safe==false)
{cout<<"Initial state is unsafe";return 0; }
cout<<"Enter the index of process and its request:";
int index,request[3];
cin>>index>>request[0]>>request[1]>>request[2];
if(need[index][0]<request[0] || need[index][1]<request[1] || need[index][2]<request[2])
{cout<<index<<" is requesting for instances more than it claimed so request denied";return 0; }
allocate[index][0]+=request[0];
allocate[index][1]+=request[1];
allocate[index][2]+=request[2];
need[index][0]-=request[0];
```

```
    need[index][1]-=request[1];
    need[index][2]-=request[2];
    available[0]-=request[0];
    available[1]-=request[1];
    available[2]-=request[2];
    safe=banker(available,allocate,need,n);
    if(safe==true)
    {cout<<"Safe state after granting request";}
    else
    {cout<<"Request cannot be granted";}
}
```




## PAGE REPLACEMENT FIFO

```
#include<queue>
#include<iostream>
#include<map>
using namespace std;
void print(queue<int>q)
{
    while(!q.empty())
    {cout<<q.front()<<" ";q.pop();}
```

```
    cout<<endl;
}
int fcfs(int n,int pages[])
{
    int page_fault=0;
    queue<int>q;
    map<int,bool>present;
    for(int i=0;pages[i]!=-1;i++)
    {
        if(present[pages[i]]==false)
        {
            page_fault++;
            if(q.size()<n){q.push(pages[i]);present[pages[i]]=true;}
            else
            {
                present[q.front()]=false;
                q.pop();
                q.push(pages[i]);
                present[pages[i]]=true;
            }
        }
        print(q);
    }
    return page_fault;
}
int main()
{
    int n;
    cout<<"Enter the number of frames:";cin>>n;
    int pages[10000];int npage=-1;
    cout<<"Enter the pages and -1 to stop\n";
    do{
        npage++;
        cin>>pages[npage];
```

```
    }while(pages[npage]!=-1);
    cout<<endl<<"page faults:"<<fcfs(n,pages);
    return 0;
}
```

```
Enter the number of frames:4
Enter the pages and -1 to stop
1 0 2 2 1 7 6 7 0 1 2 0 3 0 4 5 1 5 2 -1
1
1 0
1 0 2
1 0 2
1 0 2
1 0 2 7
0 2 7 6
0 2 7 6
0 2 7 6
2 7 6 1
2 7 6 1
7 6 1 0
6 1 0 3
6 1 0 3
1 0 3 4
0 3 4 5
3 4 5 1
3 4 5 1
4 5 1 2

page faults:12
```

## PAGE REPLACEMENT LRU

```
#include<queue>
#include<iostream>
#include<map>
using namespace std;
void print(map<int,int>q)
{
    map<int,int>::iterator it=q.begin();
    while(it!=q.end())
    {cout<<it->first<<" ";it++;}
    cout<<endl;
}
int LRU(int n,int pages[])
```

```
{
    int page_fault=0;
    map<int,int>present;
    for(int i=0;pages[i]!=-1;i++)
    {
        if(present.find(pages[i])==present.end())
        {
            page_fault++;
            if(present.size()<n)
            {present[pages[i]]=i;}
            else
            {
                int temp=present.begin()->first;
                map<int,int>::iterator x=present.begin();
                x++;
                for(;x!=present.end();x++)
                {if(x->second<present[temp]){temp=x->first;} }
                present.erase(temp);
                present[pages[i]]=i;
            }
        }
        else
        {present[pages[i]]=i; }
        print(present);
    }
    return page_fault;
}
int main()
{
    int n;
    cout<<"Enter the number of frames:";cin>>n;
    int pages[10000];int npage=-1;
    cout<<"Enter the pages and -1 to stop\n";
    do{
```

```
    npage++;
    cin>>pages[npage];
  }while(pages[npage]!=-1);
  cout<<endl<<"page faults:"<<LRU(n,pages);
  return 0;
}
```

```
Enter the number of frames:4
Enter the pages and -1 to stop
1 0 2 2 1 7 6 7 0 1 2 0 3 0 4 5 1 5 2 -1
1
0 1
0 1 2
0 1 2
0 1 2
0 1 2 7
1 2 6 7
1 2 6 7
0 1 6 7
0 1 6 7
0 1 2 7
0 1 2 7
0 1 2 3
0 1 2 3
0 2 3 4
0 3 4 5
0 1 4 5
0 1 4 5
1 2 4 5

page faults:12
```

## PAGE REPLACEMENT OPTIMAL

```
#include<queue>
#include<iostream>
#include<map>
using namespace std;
void print(map<int,int>q)
{
  map<int,int>::iterator it=q.begin();
  while(it!=q.end())
  {cout<<it->first<<" ";it++;}
  cout<<endl;
}
```

```
int optimal(int n,int pages[])
{
    int page_fault=0;
    map<int,int>present;
    for(int i=0;pages[i]!=-1;i++)
    {
        if(present.find(pages[i])==present.end())
        {
            page_fault++;
            if(present.size()<n)
            {
                present[pages[i]]=INT_MAX;
            }
            else
            {
                for(int j=i+1;pages[j]!=-1;j++)
                {
                    if(present.find(pages[j])!=present.end())
                    {
                        if(present[pages[j]]==INT_MAX)
                        {present[pages[j]]=j;}
                    }
                }
                int temp=present.begin()->first;int ans=present[temp];
                present[temp]=INT_MAX;
                map<int,int>::iterator it=present.begin();it++;
                for(;it!=present.end();it++)
                {
                    if(it->second>ans){temp=it->first;ans=it->second;}
                    present[it->first]=INT_MAX;
                }
                present.erase(temp);
                present[pages[i]]=INT_MAX;
            }
```

```
        }
        print(present);
    }
    return page_fault;
}

int main()
{
    int n;
    cout<<"Enter the number of frames:";cin>>n;
    int pages[10000];int npage=-1;
    cout<<"Enter the pages and -1 to stop\n";
    do{
        npage++;
        cin>>pages[npage];
    }while(pages[npage]!=-1);
    cout<<endl<<"page faults:"<<optimal(n,pages);
    return 0;
}
```

```
"C:\sem 4 os\page_replacement_optimal.exe"
Enter the number of frames:4
Enter the pages and -1 to stop
1 0 2 2 1 7 6 7 0 1 2 0 3 0 4 5 1 5 2 -1
1
0 1
0 1 2
0 1 2
0 1 2
0 1 2 7
0 1 6 7
0 1 6 7
0 1 6 7
0 1 6 7
0 1 2 7
0 1 2 7
0 1 2 3
0 1 2 3
1 2 3 4
1 2 4 5
1 2 4 5
1 2 4 5
1 2 4 5

page faults:9
```

# FILE MANAGEMENT

```
#include<fstream>
#include<iostream>
#include<string>
using namespace std;
class stud
{
   int age;string name;
public:
   stud(int age,string name)
   {
      this->age=age;
      this->name=name;
   }
   void print()
   {cout<<" | "<<this->age<<" "<<this->name<<" | ";}};
int main()
{
   cout<<"Binary files\n";
   fstream a;
   cout<<"Appending into the file name temp.dat\n";
   a.open("temp.dat",ios::binary|ios::app);
   cout<<"Enter the age and name of the student:";int age;string name;cin>>age>>name;
   stud ex(age,name);
   a.write((char*)&ex,sizeof(ex));
   cout<<"Closing file after appending\n";
   a.close();
   cout<<"Reading from file temp.dat\n\n";
   a.open("temp.dat",ios::binary|ios::in);
   while(a.read((char*)&ex,sizeof(ex)))
   {ex.print();}
   cout<<"\n\nClosing file";
   a.close();
   return 0;}
```

```
Binary files
Appending into the file name temp.dat
Enter the age and name of the student:20 Divyanshi
Closing file after appending
Reading from file temp.dat

 | 20 Rahul |  | 20 Sachin |  | 18 Shailja |  | 20 Akhil |  | 20 Divyanshi |

Closing file
```

## DISK SCHEDULING FCFS

```
#include<cmath>

#include<iostream>

using namespace std;


int main()

{

   int cur,s,ans=0;

   cout<<"Enter size of queue:";cin>>s;

   int arr[s];

   cout<<"Enter the requests values\n";

   for(int i=0;i<s;i++){cin>>arr[i];}

   cout<<"Enter the current pointer position:";cin>>cur;

   for(int i=0;i<s;i++)

   {

      ans+=abs(arr[i]-cur);

      cur=arr[i];

   }

   cout<<"\nCylinders used:"<<ans;

}
```

```
Enter size of queue:8
Enter the requests values
98 183 37 122 14 124 65 67
Enter the current pointer position:53

Cylinders used:640
```

# DISK SCHEDULING SSTF

```
#include<cmath>
#include<iostream>
using namespace std;
int main()
{
   int cur,s,ans=0;
   cout<<"Enter size of queue:";cin>>s;
   int arr[s];
   cout<<"Enter the requests values\n";
   for(int i=0;i<s;i++){cin>>arr[i];}
   cout<<"Enter the current pointer position:";cin>>cur;
   bool next=true;
   while(next)
   {
      next=false;int mini=-1;
      for(int i=0;i<s;i++)
      {
         if(arr[i]==-1){continue;}
         if((mini==-1) || (abs(cur-arr[mini])>abs(cur-arr[i])))
         {mini=i; next=true; }
      }
      if(next==true)
      {ans+=abs(cur-arr[mini]);cur=arr[mini];arr[mini]=-1;}
   }
   cout<<"\nCylinders used:"<<ans;
}
```

```
Enter size of queue:8
Enter the requests values
98 183 37 122 14 124 65 67
Enter the current pointer position:53

Cylinders used:236
```

# DISK SCHEDULING SCAN

```
#include<cmath>

#include<iostream>

#include<map>

using namespace std;

int main()

{

   cout<<"**************ASSUSMPTIONS********************\nhead of disk is at 0 and end is at
199\n*************************************************************\n\n";

   int cur,s,ans=0;

   cout<<"Enter size of queue:";cin>>s;

   map<int,bool>arr;

   cout<<"Enter the requests values\n";

   for(int i=0;i<s;i++){int temp;cin>>temp;arr[temp]=true;}

   cout<<"Enter the current pointer position:";cin>>cur;

   while(arr.size()>0)

   {

      for(;cur>=0 && arr.size()>0;cur--)

      {

         if(arr.find(cur)!=arr.end()){arr.erase(cur);}

         if(arr.size()==0){break;}

         if(cur>0)

         {ans++;}

      }

      cur=0;

      if(arr.size()==0){break;}

      for(;cur<=199 && arr.size()>0;cur++)

      {

         if(arr.find(cur)!=arr.end()){arr.erase(cur);}

         if(arr.size()==0){break;}

         if(cur<199)

         {ans++;}

      }

      cur=199;
```

```
    }
    cout<<"\nCylinders used:"<<ans;
}
```

```
**************ASSUSMPTIONS********************
head of disk is at 0 and end is at 199
**********************************************

Enter size of queue:8
Enter the requests values
176 79 34 60 92 11 41 114
Enter the current pointer position:50

Cylinders used:226
```

## DISK SCHEDULING C-SCAN

```
#include<cmath>
#include<iostream>
#include<map>
using namespace std;
int main()
{
    cout<<"**************ASSUSMPTIONS********************\nhead of disk is at 0 and end is at
199\n*************************************************************\n\n";
    int cur,s,ans=0;
    cout<<"Enter size of queue:";cin>>s;
    map<int,bool>arr;
    cout<<"Enter the requests values\n";
    for(int i=0;i<s;i++){int temp;cin>>temp;arr[temp]=true;}
    cout<<"Enter the current pointer position:";cin>>cur;
    while(arr.size()>0)
    {
        for(;cur<=199 && arr.size()>0;cur++)
        {
            if(arr.find(cur)!=arr.end()){arr.erase(cur);}
            if(arr.size()==0){break;}
            if(cur<199)
```

```
        {ans++;}
      }
      if(arr.size()==0){break;}
      ans+=199;
      cur=0;
    }
    cout<<"\nCylinders used:"<<ans;
}
```

```
*************ASSUSMPTIONS****************
head of disk is at 0 and end is at 199
*****************************************

Enter size of queue:8
Enter the requests values
176 79 34 60 92 11 41 114
Enter the current pointer position:50

Cylinders used:389
```

## DISK SCHEDULING LOOK

```
#include<cmath>
#include<iostream>
#include<map>
using namespace std;

int main()
{
    cout<<"**************ASSUSMPTIONS*******************\nhead of disk is at 0 and end is at 199\n*************************************************************\n\n";
    int cur,s,ans=0,big=INT_MIN,small=INT_MAX;
    cout<<"Enter size of queue:";cin>>s;
    map<int,bool>arr;
    cout<<"Enter the requests values\n";
    for(int i=0;i<s;i++)
    {
        int temp;cin>>temp;arr[temp]=true;
        if(temp>big){big=temp;}
```

```
        if(temp<small){small=temp;}
    }
    cout<<"Enter the current pointer position:";cin>>cur;
    while(arr.size()>0)
    {
        for(;cur<=big && arr.size()>0;cur++)
        {
            if(arr.find(cur)!=arr.end()){arr.erase(cur);}
            if(arr.size()==0){break;}
            if(cur<big)
            {ans++;}
        }
        cur=big;
        if(arr.size()==0){break;}
        for(;cur>=small && arr.size()>0;cur--)
        {
            if(arr.find(cur)!=arr.end()){arr.erase(cur);}
            if(arr.size()==0){break;}
            if(cur>small)
            {ans++;}
        }
        cur=small;
    }
    cout<<"\nCylinders used:"<<ans;
}
```

```
**************ASSUSMPTIONS*******************
head of disk is at 0 and end is at 199
*********************************************

Enter size of queue:8
Enter the requests values
176 79 34 60 92 11 41 114
Enter the current pointer position:50

Cylinders used:291
```

# DISK SCHEDULING C-LOOK

```
#include<cmath>

#include<iostream>

#include<map>

using namespace std;


int main()

{

   cout<<"**************ASSUSMPTIONS********************\nhead of disk is at 0 and end is at
199\n*************************************************************\n\n";

   int cur,s,ans=0,big=INT_MIN,small=INT_MAX;

   cout<<"Enter size of queue:";cin>>s;

   map<int,bool>arr;

   cout<<"Enter the requests values\n";

   for(int i=0;i<s;i++)

   {

      int temp;cin>>temp;arr[temp]=true;

      if(temp>big){big=temp;}

      if(temp<small){small=temp;}

   }

   cout<<"Enter the current pointer position:";cin>>cur;

   while(arr.size()>0)

   {

      for(;cur<=big && arr.size()>0;cur++)

      {

         if(arr.find(cur)!=arr.end()){arr.erase(cur);}

         if(arr.size()==0){break;}

         if(cur<big)

         {ans++;}

      }

      if(arr.size()==0){break;}

      cur=small;

      ans+=(big-small);

   }
```

```
    cout<<"\nCylinders used:"<<ans;

}
```

```
**************ASSUSMPTIONS*******************
head of disk is at 0 and end is at 199
**********************************************

Enter size of queue:8
Enter the requests values
176 79 34 60 92 11 41 114
Enter the current pointer position:50

Cylinders used:321
```