FCFS----------

```
#include<iostream>
using namespace std;
#define max 10
int main(){

    int P[max],CT[max],AT[max],BT[max],TAT[max],WT[max];
    float AvgWT=0, AvgTAT=0, Pno;

    cout<<"\n ---- ---- MENU ---- ----\n";
    cout<<"Enter number of processes: ";
    cin>>Pno;
    for(int i=0;i<Pno;i++){
        P[i]=i+1;
        cout<<"Enter Arrival time of P"<<i+1;
        cin>>AT[i];
        cout<<"Enter Burst time of P"<<i+1;
        cin>>BT[i];
    }

    cout<<"\nProcess No.\tArrival Time\tBurst Time";
    for(int i=0;i<Pno;i++){
        cout<<"\n"<<P[i]<<"\t\t"<<AT[i]<<"\t\t"<<BT[i];
    }

    for(int i=0;i<Pno;i++){
        for(int j=i+1;j<Pno;j++){
            if(AT[i]>AT[j]){
                swap(AT[i],AT[j]);
                swap(BT[i],BT[j]);
                swap(P[i],P[j]);
            }
        }
    }
    CT[0] = 0;
    for(int i=0 ;i<Pno;i++){
```

```
        CT[i+1] = CT[i]+ BT[i];
        TAT[i]=CT[i+1]-AT[i];
        WT[i]=TAT[i]-BT[i];
    }

    cout<<"\n --------  GANTT CHART  --------\n";
    for(int i=0;i<Pno;i++){
        cout<<"|  P"<<P[i]<<"\t";
    }
    cout<<"|\n";
    for(int i=0;i<=Pno;i++){
        cout<<CT[i]<<"\t";
    }

    for(int i=0;i<Pno;i++){
        cout<<"\nWaiting time for P"<<P[i]<<": "<<WT[i];
        AvgWT+=WT[i];
    }
    cout<<"\nAvg WT:   "<<AvgWT/Pno;
    for(int i=0;i<Pno;i++){
        cout<<"\nTurn Around time for P"<<P[i]<<": "<<TAT[i];
        AvgTAT+=TAT[i];
    }
    cout<<"\nAvg TAT:   "<<AvgTAT/Pno;

    return 0;
}
```

Priority ----------

```
#include<iostream>
using namespace std;
#define max 10
int main(){

    int
P[max],CT[max],AT[max],BT[max],TAT[max],WT[max],is_comp[max],ST[max],Priority[max],ind
ex,curr_time=0,completed =0;
    float AvgWT=0, AvgTAT=0, Pno;

    int x=0;
```

```
cout<<"\n ---- ---- MENU ---- ----\n";
cout<<"Enter number of processes: ";
cin>>Pno;
for(int i=0;i<Pno;i++){
    P[i]=i+1;
    cout<<"Enter Arrival time of P"<<i+1;
    cin>>AT[i];
    cout<<"Enter Burst time of P"<<i+1;
    cin>>BT[i];
    cout<<"Enter Priority of P"<<i+1;
    cin>>Priority[i];
}

cout<<"\nPriority\tProcess No.\tArrival Time\tBurst Time";
for(int i=0;i<Pno;i++){
    cout<<"\n"<<Priority[i]<<"\t\t"<<P[i]<<"\t\t"<<AT[i]<<"\t\t"<<BT[i];
}

for(int i=0;i<Pno;i++){
    is_comp[i]=0;
}

cout<<"\n --------  GANTT CHART  --------\n";
while(completed<Pno){
    int maxP = -1;
    for(int i=0;i<Pno;i++){
        if(AT[i]<=curr_time && is_comp[i] == 0){
            if(Priority[i]>maxP){
                maxP = Priority[i];
                index = i;
            }
            if(Priority[i]==maxP){
                if(AT[i]<AT[index]){
                    maxP = Priority[i];
                    index = i;
                }
            }
        }
    }
```

```
        if(maxP == -1){
            curr_time++;
        }
        else{
            ST[index] = curr_time;
            CT[x] = ST[index] + BT[index];
            TAT[index] = CT[x] - AT[index];
            cout<<"curr --->"<<curr_time;
            WT[index] = TAT[index] - BT[index];
            is_comp[index] = 1;
            completed++;
            curr_time=CT[x];

            x++;
            cout<<"|  P"<<P[index]<<"\t";
        }
    }
    cout<<"|\n0\t";
    for(int i=0;i<Pno;i++){
        cout<<CT[i]<<"\t";
    }

    for(int i=0;i<Pno;i++){
        cout<<"\nWaiting time for P"<<P[i]<<": "<<WT[i];
        AvgWT+=WT[i];
    }
    cout<<"\nAvg WT:   "<<AvgWT/Pno;
    for(int i=0;i<Pno;i++){
        cout<<"\nTurn Around time for P"<<P[i]<<": "<<TAT[i];
        AvgTAT+=TAT[i];
    }
    cout<<"\nAvg TAT:   "<<AvgTAT/Pno;

    return 0;
}
RR ----------------------------------------------------------------------------------------------------
#include<iostream>
using namespace std;
#define max 20
```

```
int main(){

    int
P[max],CT[max],AT[max],BT[max],TAT[max],WT[max],RT[max],TQ,time=0,curr_time=0,compl
eted =0;
    float AvgWT=0, AvgTAT=0, Pno;

    int exe_ord[max],exe_time[max],exe_index=0;

    int x=0;
    cout<<"\n ---- ---- MENU ---- ----\n";
    cout<<"Enter number of processes: ";
    cin>>Pno;
    cout<<"Enter Time Quantum: ";
    cin>>TQ;
    for(int i=0;i<Pno;i++){
        P[i]=i+1;
        cout<<"Enter Arrival time of P"<<i+1;
        cin>>AT[i];
        cout<<"Enter Burst time of P"<<i+1;
        cin>>BT[i];
        RT[i]=BT[i];
    }

    cout<<"\nProcess No.\tArrival Time\tBurst Time";
    for(int i=0;i<Pno;i++){
        cout<<"\n"<<P[i]<<"\t\t"<<AT[i]<<"\t\t"<<BT[i];
    }


    while(completed<Pno){
        for(int i=0;i<Pno;i++){
            if(AT[i]<=curr_time && RT[i]>0 ){
                time = min(RT[i],TQ);
                curr_time+=time;
                RT[i]-=time;
                exe_ord[exe_index]=P[i];
                exe_time[exe_index]=curr_time;
                exe_index++;
```

```
            if(RT[i]==0){
                CT[i]= curr_time;
                TAT[i]= CT[i]-AT[i];
                WT[i]= TAT[i] - BT[i];
                completed++;
            }
        }
    }
}

    cout<<"\n --------  GANTT CHART  --------\n";
    for(int i=0;i<exe_index;i++){
        cout<<"|  P"<<exe_ord[i]<<"\t";
    }
    cout<<"|\n0\t";
    for(int i=0;i<exe_index;i++){
        cout<<exe_time[i]<<"\t";
    }

    for(int i=0;i<Pno;i++){
        cout<<"\nWaiting time for P"<<P[i]<<": "<<WT[i];
        AvgWT+=WT[i];
    }
    cout<<"\nAvg WT:   "<<AvgWT/Pno;
    for(int i=0;i<Pno;i++){
        cout<<"\nTurn Around time for P"<<P[i]<<": "<<TAT[i];
        AvgTAT+=TAT[i];
    }
    cout<<"\nAvg TAT:   "<<AvgTAT/Pno;

    return 0;
}
```

SJF ----------

```
#include<iostream>
using namespace std;
#define max 20
int main(){
```

```
    int
P[max],CT[max],AT[max],BT[max],TAT[max],WT[max],RT[max],time,curr_time=0,completed
=0;
    float AvgWT=0, AvgTAT=0, Pno;

    int exe_ord[max],exe_time[max],exe_index=0;

    int x=0;
    cout<<"\n ---- ---- MENU ---- ----\n";
    cout<<"Enter number of processes: ";
    cin>>Pno;
    for(int i=0;i<Pno;i++){
        P[i]=i+1;
        cout<<"Enter Arrival time of P"<<i+1;
        cin>>AT[i];
        cout<<"Enter Burst time of P"<<i+1;
        cin>>BT[i];
        RT[i]=BT[i];
    }

    cout<<"\nProcess No.\tArrival Time\tBurst Time";
    for(int i=0;i<Pno;i++){
        cout<<"\n"<<P[i]<<"\t\t"<<AT[i]<<"\t\t"<<BT[i];
    }


    while(completed<Pno){
        int shortest_time = 99;
        int shortest_ind = -1;
        for(int i=0;i<Pno;i++){
            if(AT[i]<=curr_time && RT[i]>0 && RT[i]<shortest_time){
                shortest_ind = i;
                shortest_time = RT[i];
            }
        }

        if(shortest_ind != exe_ord[exe_index-1]){
            exe_ord[exe_index]=shortest_ind;
            exe_time[exe_index]=curr_time;
```

```
        exe_index++;
    }

    if(shortest_ind==-1){
        curr_time++;
    }
    else{
        RT[shortest_ind]--;
        curr_time++;
        if(RT[shortest_ind]==0){
            CT[shortest_ind] = curr_time;
            TAT[shortest_ind] = CT[shortest_ind] - AT[shortest_ind];
            WT[shortest_ind] = TAT[shortest_ind] - BT[shortest_ind];
            completed++;
        }
    }
    time = curr_time;
}

cout<<"\n --------  GANTT CHART  --------\n";
for(int i=0;i<exe_index;i++){
    cout<<"|  P"<<exe_ord[i]<<"\t";
}
cout<<"|\n";
for(int i=0;i<exe_index;i++){
    cout<<exe_time[i]<<"\t";
}
cout<<time;

for(int i=0;i<Pno;i++){
    cout<<"\nWaiting time for P"<<P[i]<<": "<<WT[i];
    AvgWT+=WT[i];
}
cout<<"\nAvg WT:   "<<AvgWT/Pno;
for(int i=0;i<Pno;i++){
    cout<<"\nTurn Around time for P"<<P[i]<<": "<<TAT[i];
    AvgTAT+=TAT[i];
}
cout<<"\nAvg TAT:   "<<AvgTAT/Pno;
```

```
    return 0;
}
```

Best fit ----------

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

int main(){

    int m, n;

    cout<<"Enter Number of Blocks :";
    cin>>n;
    int blockSize[n];
    for(int i = 0; i<n;i++){
        cout<<"Enter size of BlockSize ["<<i<<"]";
        cin>>blockSize[i];
    }

    cout<<"Enter Number of Processes :";
    cin>>m;
    int processSize[m];
    for(int i = 0; i<m;i++){
        cout<<"Enter size of Process ["<<i<<"]";
        cin>>processSize[i];
    }

    int address[n];
    address[0]=0;
    for(int i=1;i<n;i++){
        address[i]=address[i-1]+blockSize[i-1];
    }
    int total = address[n-1]+blockSize[n-1];

    int allocation[m];
    for(int i=0;i<m;i++)
        allocation[i]=-1;
```

```
for(int i=0;i<m;i++){
    int bestInd = -1;
    for(int j=0;j<n;j++){
        if(blockSize[j]>=processSize[i]){
            if(bestInd == -1 || blockSize[bestInd]>blockSize[j]){
                bestInd = j;
            }
        }
    }
    if(bestInd!=-1){
        allocation[i]=bestInd;
        blockSize[bestInd]-=processSize[i];
    }


}

cout<<"\nProcesses\tProcess Size\tBlock No.\t";
for(int i =0;i<m;i++){
    cout<<"\nP"<<i<<"\t\t"<<processSize[i]<<"\t\t";
    if(allocation[i]!=-1)
        cout<<allocation[i];
    else
        cout<<"Not allocated";
}

cout<<"\n\nBlock No.\tAllocation\tAddress";
for(int i=0;i<n;i++){
    cout<<"\n"<<i<<"\t\t";
    for(int x=0;x<m;x++){
        if(allocation[x]!=-1){
            if(allocation[x]==i)
                cout<<"p"<<x;
        }
        else{
            break;
        }
    }
    cout<<"\t\t";
```

```
        cout<<address[i];
    }

    cout<<"\n total: "<<total;
    float used=0;
    for (int i=0;i<m;i++){
        if(allocation[i]!=-1){
            used += processSize[i];
        }
    }
    cout<<"\nused: "<<used;
    cout<<"\nMemory utilization: "<<used/total;

    return 0;
}
```

First fit ----------

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

int main(){

    int m, n;

    cout<<"Enter Number of Blocks :";
    cin>>n;
    int blockSize[n];
    for(int i = 0; i<n;i++){
        cout<<"Enter size of BlockSize ["<<i<<"]";
        cin>>blockSize[i];
    }

    cout<<"Enter Number of Processes :";
    cin>>m;
    int processSize[m];
    for(int i = 0; i<m;i++){
        cout<<"Enter size of Process ["<<i<<"]";
        cin>>processSize[i];
    }
```

```
int address[n];
address[0]=0;
for(int i=1;i<n;i++){
    address[i]=address[i-1]+blockSize[i-1];
}
int total = address[n-1]+blockSize[n-1];

int allocation[m];
for(int i=0;i<m;i++)
    allocation[i]=-1;

for(int i=0;i<m;i++){
    for(int j=0;j<n;j++){
        if(blockSize[j]>=processSize[i]){
            allocation[i]=j;
            blockSize[j] -= processSize[i];
            break;
        }
    }
}

cout<<"\nProcesses\tProcess Size\tBlock No.\t";
for(int i =0;i<m;i++){
    cout<<"\nP"<<i<<"\t\t"<<processSize[i]<<"\t\t";
    if(allocation[i]!=-1)
        cout<<allocation[i];
    else
        cout<<"Not allocated";
}

cout<<"\n\nBlock No.\tAllocation\tAddress";
for(int i=0;i<n;i++){
    cout<<"\n"<<i<<"\t\t";
    for(int x=0;x<m;x++){
        if(allocation[x]!=-1){
            if(allocation[x]==i)
                cout<<"p"<<x;
        }
```

```
            else{
                break;
            }
        }
        cout<<"\t\t";
        cout<<address[i];
    }

    cout<<"\n total: "<<total;
    float used=0;
    for (int i=0;i<m;i++){
        if(allocation[i]!=-1){
            used += processSize[i];
        }
    }
    cout<<"\nused: "<<used;
    cout<<"\nMemory utilization: "<<used/total;

    return 0;
}
```

Next fit ----------

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

int main(){

    int m, n;

    cout<<"Enter Number of Blocks :";
    cin>>n;
    int blockSize[n];
    for(int i = 0; i<n;i++){
        cout<<"Enter size of BlockSize ["<<i<<"]";
        cin>>blockSize[i];
    }

    cout<<"Enter Number of Processes :";
    cin>>m;
```

```
int processSize[m];
for(int i = 0; i<m;i++){
    cout<<"Enter size of Process ["<<i<<"]";
    cin>>processSize[i];
}

int tbs[n];
for(int i=0;i<n;i++){
    tbs[i] = blockSize[i];
}

int address[n];
address[0]=0;
for(int i=1;i<n;i++){
    address[i]=address[i-1]+blockSize[i-1];
}
int total = address[n-1]+blockSize[n-1];

int allocation[m];
for(int i=0;i<m;i++)
    allocation[i]=-1;

int ptr = -1;
for(int i=0;i<m;i++){
    for(int j=0;j<n;j++){
        ptr = (ptr + 1) % n;
        if(blockSize[ptr]>processSize[i]){
            allocation[i]=ptr;
            blockSize[ptr]-=processSize[i];
            break;
        }
    }
}

cout<<"\nProcesses\tProcess Size\tBlock No.\t";
for(int i =0;i<m;i++){
    cout<<"\nP"<<i<<"\t\t"<<processSize[i]<<"\t\t";
    if(allocation[i]!=-1)
        cout<<allocation[i];
```

```
        else
            cout<<"Not allocated";
    }

    cout<<"\n\nBlock No.\tAllocation\tAddress";
    for(int i=0;i<n;i++){
        cout<<"\n"<<i<<"\t\t";
        for(int x=0;x<m;x++){
            if(allocation[x]!=-1){
                if(allocation[x]==i)
                    cout<<"p"<<x;
            }
            else{
                cout<<"["<<tbs[i]<<"]";
                break;
            }
        }
        cout<<"\t\t";
        cout<<address[i];
    }

    cout<<"\n total: "<<total;
    float used=0;
    for (int i=0;i<m;i++){
        if(allocation[i]!=-1){
            used += processSize[i];
        }
    }
    cout<<"\nused: "<<used;
    cout<<"\nMemory utilization: "<<used/total;

    return 0;
}
```

Worst fit ----------

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

int main(){
```

```
int m, n;

cout<<"Enter Number of Blocks :";
cin>>n;
int blockSize[n];
for(int i = 0; i<n;i++){
    cout<<"Enter size of BlockSize ["<<i<<"]";
    cin>>blockSize[i];
}

cout<<"Enter Number of Processes :";
cin>>m;
int processSize[m];
for(int i = 0; i<m;i++){
    cout<<"Enter size of Process ["<<i<<"]";
    cin>>processSize[i];
}

int address[n];
address[0]=0;
for(int i=1;i<n;i++){
    address[i]=address[i-1]+blockSize[i-1];
}
int total = address[n-1]+blockSize[n-1];

int allocation[m];
for(int i=0;i<m;i++)
    allocation[i]=-1;

for(int i=0;i<m;i++){
    int worstInd = -1;
    for(int j=0;j<n;j++){
        if(blockSize[j]>=processSize[i]){
            if(worstInd == -1 || blockSize[worstInd]<blockSize[j]){
                worstInd = j;
            }
        }
    }
    if(worstInd!=-1){
```

```
            allocation[i]=worstInd;
            blockSize[worstInd]-=processSize[i];
            cout<<blockSize[worstInd];
        }


}

cout<<"\nProcesses\tProcess Size\tBlock No.\t";
for(int i =0;i<m;i++){
    cout<<"\nP"<<i<<"\t\t"<<processSize[i]<<"\t\t";
    if(allocation[i]!=-1)
        cout<<allocation[i];
    else
        cout<<"Not allocated";
}

cout<<"\n\nBlock No.\tAllocation\tAddress";
for(int i=0;i<n;i++){
    cout<<"\n"<<i<<"\t\t";
    for(int x=0;x<m;x++){
        if(allocation[x]!=-1){
            if(allocation[x]==i)
                cout<<"p"<<x;
        }
        else{
            break;
        }
    }
    cout<<"\t\t";
    cout<<address[i];
}

cout<<"\n total: "<<total;
float used=0;
for (int i=0;i<m;i++){
    if(allocation[i]!=-1){
        used += processSize[i];
    }
```

```
    }
    cout<<"\nused: "<<used;
    cout<<"\nMemory utilization: "<<used/total;

    return 0;
}
```

Semaphore ----------

++

```
#include<iostream>
using namespace std;
#define max 5

void signal(int &x){
    x++;
}

void wait(int &x){
    if(x>0)
        x--;
}

class Semaphore{
public:
    int buffer[max];
    int empty = max;
    int full = 0;
    int mutex = 1;

    void producer(){
        cout<<"Empty: "<<empty<<"  Full: "<<full<<"  Mutex: "<<mutex;
        if (empty!=0 && mutex == 1){
            wait(empty);
            wait(mutex);
```

```
            cout<<"Enter item to produce: ";
            cin>>buffer[full];
            signal(mutex);
            signal(full);
        }
    }

    void consumer(){
        cout<<"Empty: "<<empty<<"  Full: "<<full<<"  Mutex: "<<mutex;
        if (full!=0 && mutex == 1){
            wait(full);
            wait(mutex);
            cout<<"Item Consumed is: "<<buffer[full];
            signal(mutex);
            signal(empty);
        }
    }
};

int main(){
    Semaphore s;
    int ch;
    while(true){
    cout<<"\n ----- ----- MENU ----- ----- \n";
    cout<<"1.Producer\n2.Consumer\n3.Exit\nEnter choice: ";
    cin>>ch;
    if (ch ==1)
        s.producer();
    else if (ch ==2)
        s.consumer();
    else if (ch ==3)
        break;
    else
        cout<<"\nInvalid choice\n";
    }
    return 0;
}
```

==========

JNI

B1.java

```
import java.io.*;
import java.util.*;
class B1 {
   static {
        System.loadLibrary("b1");
   }
   private native int add(int a, int b);
   public static void main(String[] args) {
   Scanner sc=new Scanner(System.in);
   int a, b,ch;
   System.out.println("\nEnter value of a : ");
   a = sc.nextInt();
   System.out.println("\nEnter value of b : ");
   b = sc.nextInt();
   do
   {
      System.out.println("\nENTER YOUR CHOICE : ");
      ch = sc.nextInt();
      switch(ch)
      {
         case 1 : new B1().add(a,b);
             break;
         default : System.out.println("Your choice is wrong.");
      }
   }while(ch<2);
    }
}
```

B1.c

```
#include <jni.h>
#include <stdio.h>
#include "B1.h"
JNIEXPORT int JNICALL Java_B1_add(JNIEnv *env, jobject obj,   jint a, jint b)
{
   printf("\n%d + %d = %d\n",a,b,(a+b));
   return 0;
}
```

Output ---------->

Steps to execute the program ---------->

```
 javac -h . B1.java
(base) admin1@408-21:~$ javac B1.java
(base) admin1@408-21:~$ javah B1
(base) admin1@408-21:~$  gcc -fPIC -I"$JAVA_HOME/include" -
I"$JAVA_HOME/include/linux" -shared -o libb1.so B1.c
(base) admin1@408-21:~$ java -Djava.library.path=. B1

Enter value of a :
2

Enter value of b :
3

ENTER YOUR CHOICE :
1

2 + 3 = 5

ENTER YOUR CHOICE :
```