Thread_code:

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#include<sys/types.h>

int count=0;
int a[10];

void* producer()
{
    while(1)
    {
        int x=rand()%100;
        a[count]=x;
        printf("Produced Item : %d\n",x);
        count=count+1;
        if(count==10)
        {
            sleep(1);
        }
    }
}

void* consumer()
{
    while(1)
    {
        if(count==0)
        {
            sleep(1);
        }
        count=count-1;
        int y=a[count];
        printf("Consumed Item : %d\n",y);
    }
}


int main()
{
    pthread_t t1,t2;
    pthread_create(&t1,NULL,&producer,NULL);
    pthread_create(&t2,NULL,&consumer,NULL);
```

```
    pthread_join(t1,NULL);
    pthread_join(t2,NULL);
    return 0;
}
```

Mutex_Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>
#include<pthread.h>

pthread_mutex_t m;
int count=0;
int a[10];

void* producer()
{
    while(1)
    {
        int x=rand()%100;
        pthread_mutex_lock(&m);
        a[count]=x;
        printf("Produced Item : %d\n",x);
        count++;
        pthread_mutex_unlock(&m);
        if(count==10)
        {
            sleep(1);
        }
    }
}

void* consumer()
{
    while(1)
    {
        pthread_mutex_lock(&m);
        count--;
        int y=a[count];
        printf("Consumed Item : %d\n",y);
        pthread_mutex_unlock(&m);
        if(count==0)
        {
```

```
            sleep(1);
        }
    }
}

int main()
{
    pthread_t t1,t2;
    pthread_mutex_init(&m,NULL);
    pthread_create(&t1,NULL,&producer,NULL);
    pthread_create(&t2,NULL,&consumer,NULL);
    pthread_join(t1,NULL);
    pthread_join(t2,NULL);
    pthread_mutex_destroy(&m);
    return 0;
}
```

FCFS_CODE:

```
#! /bin/bash
echo "Enter bt: "
read -a bt
n=${#bt[@]}
n=$((n-1))
wt=(0 0 0 0 0 0 0 0)
for ((i=1;i<=$n;i++));
do
     wt[i]=$((bt[i-1]+wt[i-1]))
done
total=0
for t in ${wt[@]}
do
    total=$((total+t))
done
n=$((n+1))
echo "awt"
echo "scale=3;$total/$n"|bc
```

SJF_CODE:

```
#! /bin/bash
```

```
echo "Enter bt(burst time) : "
read -a bt
bt=($(printf '%s\n' "${bt[@]}" | sort -n))
n=${#bt[@]}
n=$((n-1))
wt=(0 0 0 0 0 0 0 0)
for ((i=1;i<=$n;i++));
do
   wt[i]=$((bt[i-1]+wt[i-1]))
done
total=0
for t in ${wt[@]}
do
   total=$((total+t))
done
n=$((n+1))
echo "Average Waiting time : "
echo "scale=3;$total/$n"|bc
```

Fork_Code:

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
int main()
{
int n1=fork();
if(n1>0)
{
 printf("[parent] pid %d\n",getpid());
 sleep(20);
 int n2=fork();
 if(n2==0)
  {
    printf("[son] pid %d from [parent] pid %d \n",getpid(),getppid());
    sleep(20);
   int n5=fork();
    if(n5>0)
     {
       int n6=fork();
       if(n6==0)
        {
```

```
          printf("[son] pid %d from [parent] pid %d \n",getpid(),getppid());
          sleep(20);
        }
        sleep(20);

     }
    else if(n5==0)
     {
       printf("[son] pid %d form [parent] pid %d \n",getpid(),getppid());
       sleep(20);
     }
     sleep(20);
  }
}
else if(n1==0)
{
  printf("[son] pid %d form [parent] pid %d \n",getpid(),getppid());
  sleep(20);
 int n3=fork();
  if(n3>0)
   {
     int n4=fork();
     if(n4==0)
      {
        printf("[son] pid %d from [parent] pid %d \n",getpid(),getppid());
        sleep(20);
      }
      sleep(20);
   }
  else if(n3==0)
    {
       printf("[son] pid %d from [parent] pid %d \n",getpid(),getppid());
       sleep(20);
    }
     sleep(20);

}

return 0;
}
```