OS LAB DA- 3, 4

(ELA)

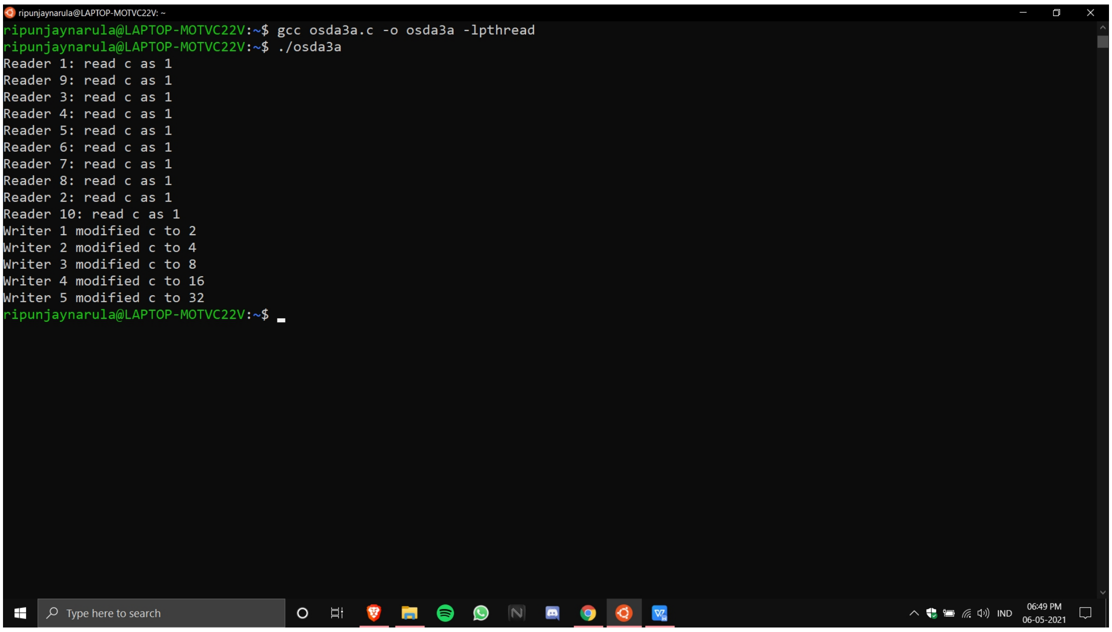
Name: **Ripunjay Narula**

Reg No.: **19BCE0470**

ASSESMENT-3

a) Implement the solution for reader – writer's problem

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

sem_t wrt;
pthread_mutex_t mutex;
int c = 1;
int numreader = 0;
void *writer(void *wno)
{
    sem_wait(&wrt);
    c = c*2;
    printf("Writer %d modified c to %d\n",(*((int *)wno)),c);
    sem_post(&wrt);

}
void *reader(void *rno)
{
    pthread_mutex_lock(&mutex);
    numreader++;
    if(numreader == 1) {
        sem_wait(&wrt);
    }
    pthread_mutex_unlock(&mutex);
    printf("Reader %d: read c as %d\n",*((int *)rno),c);
    pthread_mutex_lock(&mutex);
    numreader--;
    if(numreader == 0) {
        sem_post(&wrt);
    }
    pthread_mutex_unlock(&mutex);
}
int main()
{

    pthread_t read[10],write[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&wrt,0,1);
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    for(int i = 0; i < 10; i++) {
        pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
    }
    for(int i = 0; i < 10; i++) {
        pthread_join(read[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(write[i], NULL);
    }
    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);
    return 0;
}
```



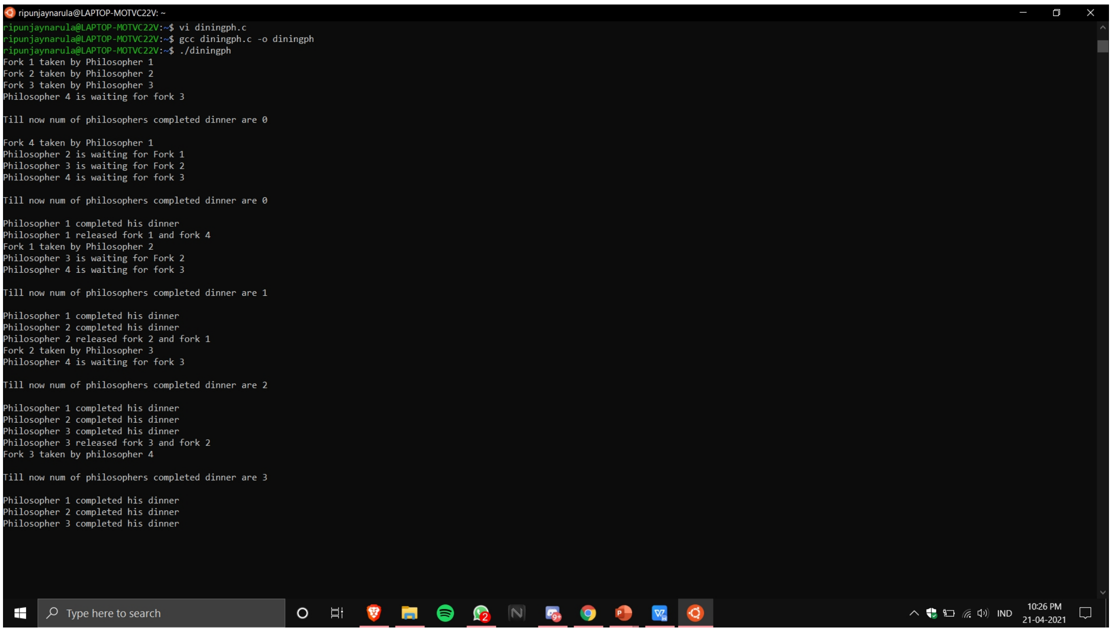
```
ripunjaynarula@LAPTOP-MOTVC22V:~$ gcc osda3a.c -o osda3a -lpthread
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./osda3a
Reader 1: read c as 1
Reader 9: read c as 1
Reader 3: read c as 1
Reader 4: read c as 1
Reader 5: read c as 1
Reader 6: read c as 1
Reader 7: read c as 1
Reader 8: read c as 1
Reader 2: read c as 1
Reader 10: read c as 1
Writer 1 modified c to 2
Writer 2 modified c to 4
Writer 3 modified c to 8
Writer 4 modified c to 16
Writer 5 modified c to 32
ripunjaynarula@LAPTOP-MOTVC22V:~$ _
```

b) Implement the solution for dining philosopher’s 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);
            }
          }
    }else{}
}

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);
       }
```

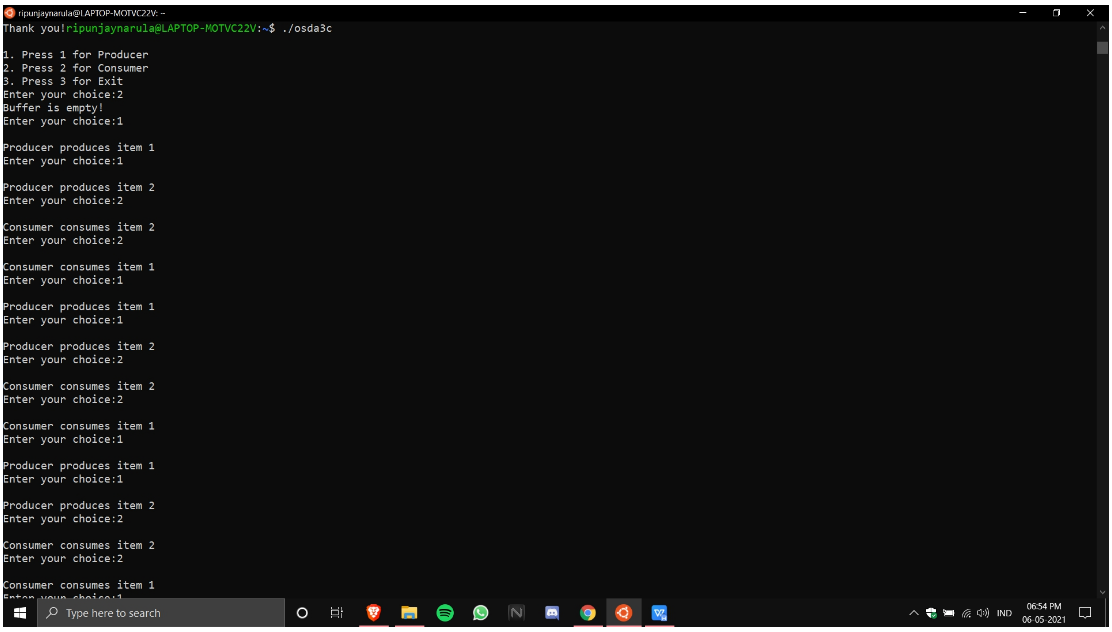
```
        return 0;
}
```



```
ripunjaynarula@LAPTOP-MOTVC22V:~$ vi diningph.c
ripunjaynarula@LAPTOP-MOTVC22V:~$ gcc diningph.c -o diningph
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./diningph
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
```

c) Implement the solution for producer consumer problem

```
#include <stdio.h>
#include <stdlib.h>

int mutex = 1;
int f = 0;
int e = 10,
x = 0;

void Producer()
{
    --mutex;
    ++f;
    --e;
    x++;
    printf("\nProducer produces item %d",x);
    ++mutex;
}
void Consumer()
{
    --mutex;
    --f;
    ++e;
    printf("\nConsumer consumes item %d",x);
    x--;
    ++mutex;
}
int main()
{
    int n, i;
    printf("\n1. Press 1 for Producer"
           "\n2. Press 2 for Consumer"
           "\n3. Press 3 for Exit");
#pragma omp critical

    for (i = 1; i > 0; i++) {

        printf("\nEnter your choice:");
        scanf("%d", &n);
        switch (n) {
        case 1:
            if ((mutex == 1) && (e != 0))
            {
                Producer();
            }
            else
            {
                printf("Buffer is full!");
            }
            break;

        case 2:
            if ((mutex == 1) && (f != 0))
            {
                Consumer();
            }
            else {
                printf("Buffer is empty!");
            }
            break;
        case 3:
            printf("Thank you!");
            exit(0);
            break;
        }
    }
}
```



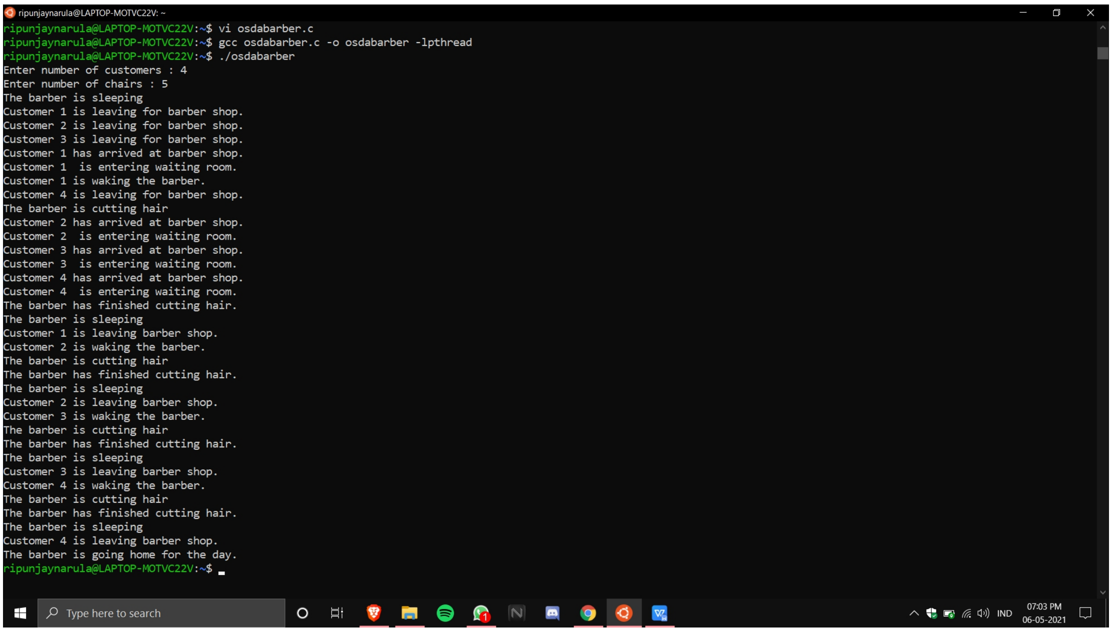
```
Thank you!ripunjaynarula@LAPTOP-MOTVC22V:~$ ./osda3c

1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice:2
Buffer is empty!
Enter your choice:1

Producer produces item 1
Enter your choice:1

Producer produces item 2
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:1

Producer produces item 1
Enter your choice:1

Producer produces item 2
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:1

Producer produces item 1
Enter your choice:1

Producer produces item 2
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:1
```

d)The analogy is based upon a hypothetical barber shop with one barber. There is a barber shop which has one barber, one barber chair, and n chairs for waiting for customers if there are any to sit on the chair.

- **If there is no customer, then the barber sleeps in his own chair.**
- **When a customer arrives, he has to wake up the barber.**
- **If there are many customers and the barber is cutting a customer's hair, then the remaining customers either wait if there are empty chairs in the waiting room or they leave if no chairs are empty.**

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#include <pthread.h>
#include <semaphore.h>
#define MAX_CUSTOMERS 5
void *customer(void *num);
void *barber(void *);
void randwait(int secs);
sem_t waitingRoom;sem_t barberChair;
sem_t barberPillow;
sem_t seatBelt;
int allDone = 0;
int main(int argc, char *argv[]) {
pthread_t btid;
pthread_t tid[MAX_CUSTOMERS];
long RandSeed;
int i, numCustomers, numChairs;
int Number[MAX_CUSTOMERS];
printf("Enter number of customers : "); scanf("%d",&numCustomers) ;
printf("Enter number of chairs : "); scanf("%d",&numChairs);
if (numCustomers > MAX_CUSTOMERS) {
printf("The maximum number of Customers is %d.\n", MAX_CUSTOMERS);
exit(-1);
}
for (i=0; i<MAX_CUSTOMERS; i++) {
Number[i] = i;
}
sem_init(&waitingRoom, 0, numChairs);
sem_init(&barberChair, 0, 1);
sem_init(&barberPillow, 0, 0);
sem_init(&seatBelt, 0, 0);
pthread_create(&btid, NULL, barber, NULL);
for (i=0; i<numCustomers; i++) {
```

```
pthread_create(&tid[i], NULL, customer, (void *)&Number[i]);
sleep(1);
}
for (i=0; i<numCustomers; i++) {
pthread_join(tid[i],NULL);
sleep(1);
}
allDone = 1;
sem_post(&barberPillow);
pthread_join(btid,NULL);
}
void *customer(void *number) {
int num = *(int *)number;printf("Customer %d is leaving for barber shop.\n", num+1);
randwait(2);
printf("Customer %d has arrived at barber shop.\n", num+1);
sem_wait(&waitingRoom);
printf("Customer %d  is entering waiting room.\n", num+1);
sem_wait(&barberChair);
sem_post(&waitingRoom);
printf("Customer %d is waking the barber.\n", num+1);
sem_post(&barberPillow);
sem_wait(&seatBelt);
sem_post(&barberChair);
printf("Customer %d is leaving barber shop.\n", num+1);
}
void *barber(void *junk) {
while (!allDone) {
printf("The barber is sleeping\n");
sem_wait(&barberPillow);
if (!allDone) {
printf("The barber is cutting hair\n");
randwait(2);
printf("The barber has finished cutting hair.\n");
sem_post(&seatBelt);
}
else {
printf("The barber is going home for the day.\n");
}
}
}
void randwait(int secs) {
int len;
len = (int) ((1 * secs) + 1);
sleep(len);
}
```



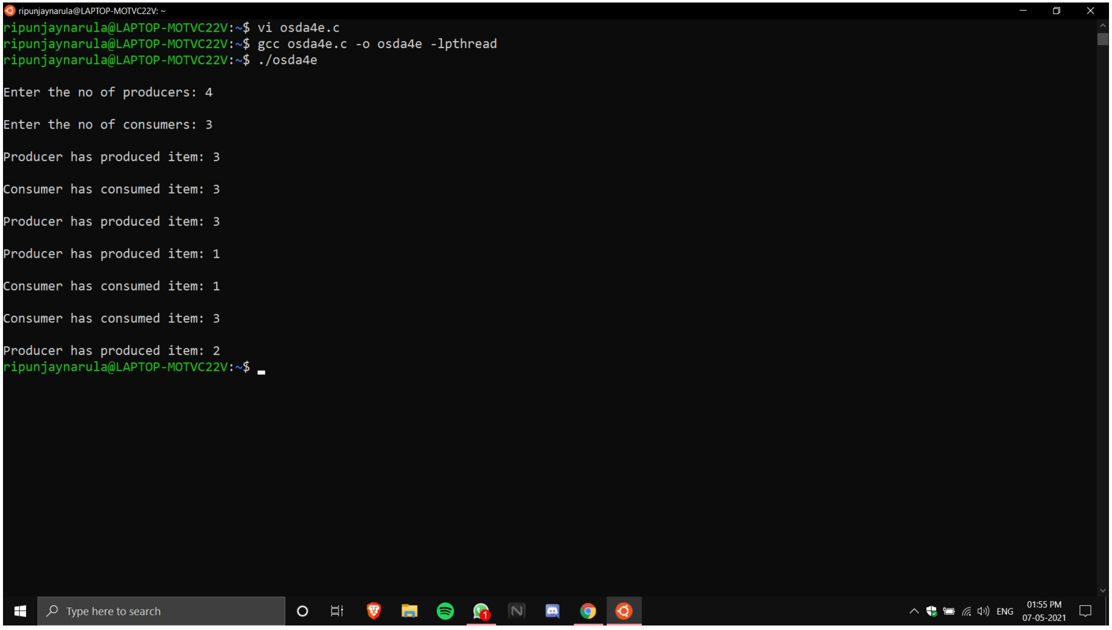
```
ripunjaynarula@LAPTOP-MOTVC22V:~$ vi osdabarber.c
ripunjaynarula@LAPTOP-MOTVC22V:~$ gcc osdabarber.c -o osdabarber -lpthread
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./osdabarber
Enter number of customers : 4
Enter number of chairs : 5
The barber is sleeping
Customer 1 is leaving for barber shop.
Customer 2 is leaving for barber shop.
Customer 3 is leaving for barber shop.
Customer 1 has arrived at barber shop.
Customer 1  is entering waiting room.
Customer 1 is waking the barber.
Customer 4 is leaving for barber shop.
The barber is cutting hair
Customer 2 has arrived at barber shop.
Customer 2  is entering waiting room.
Customer 3 has arrived at barber shop.
Customer 3  is entering waiting room.
Customer 4 has arrived at barber shop.
Customer 4  is entering waiting room.
The barber has finished cutting hair.
The barber is sleeping
Customer 1 is leaving barber shop.
Customer 2 is waking the barber.
The barber is cutting hair
The barber has finished cutting hair.
The barber is sleeping
Customer 2 is leaving barber shop.
Customer 3 is waking the barber.
The barber is cutting hair
The barber has finished cutting hair.
The barber is sleeping
Customer 3 is leaving barber shop.
Customer 4 is waking the barber.
The barber is cutting hair
The barber has finished cutting hair.
The barber is sleeping
Customer 4 is leaving barber shop.
The barber is going home for the day.
ripunjaynarula@LAPTOP-MOTVC22V:~$
```

e) A pair of processes involved in exchanging a sequence of integers. The number of integers that can be produced and consumed at a time is limited to 100. Write a Program to implement the producer and consumer problem using POSIX semaphore for the above scenario.

```
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>
#include<stdlib.h>
#define buffersize 100
pthread_mutex_t mutex;
pthread_t tidP[100],tidC[100];
sem_t full,empty;
int counter;
int buffer[buffersize];
void initialize()
{
 pthread_mutex_init(&mutex, NULL);
 sem_init(&full,1,0);
 sem_init(&empty,1,buffersize);
 counter=0;
}
void write(int item)
{
 buffer[counter++]=item;
}
int read()
{
 return(buffer[--counter]);
}
void *producer(void *param)
{
 int waittime,item;
 item=rand()%5;
 waittime=rand()%5;
 sem_wait(&empty);
 pthread_mutex_lock(&mutex);
 printf("\nProducer has produced item: %d\n",item);
 write(item);
 pthread_mutex_unlock(&mutex);
 sem_post(&full);
}
void *consumer(void * param)
{
 int waittime, item;
 waittime=rand()%5;
 sem_wait(&full);
 pthread_mutex_lock(&mutex);
 item=read();
 printf("\nConsumer has consumed item: %d\n",item);
 pthread_mutex_unlock(&mutex);
 sem_post(&empty);
}
```

```
int main()
{
 int n1,n2,i;
 initialize();
 printf("\nEnter the no of producers: ");
 scanf("%d",&n1);
 printf("\nEnter the no of consumers: ");
 scanf("%d",&n2);
 for(i=0;i<n1;i++)
 {
 pthread_create(&tidP[i],NULL,producer,NULL);
 }
 for(i=0;i<n2;i++)
 {
 pthread_create(&tidC[i],NULL,consumer,NULL);
 }
 for(i=0;i<n1;i++)
 {
 pthread_join(tidP[i],NULL);
 }
 for(i=0;i<n2;i++)
 {
 pthread_join(tidC[i],NULL);
 }
 exit(0);
```

}



```
ripunjaynarula@LAPTOP-MOTVC22V:~$ vi osda4e.c
ripunjaynarula@LAPTOP-MOTVC22V:~$ gcc osda4e.c -o osda4e -lpthread
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./osda4e

Enter the no of producers: 4

Enter the no of consumers: 3

Producer has produced item: 3

Consumer has consumed item: 3

Producer has produced item: 3

Producer has produced item: 1

Consumer has consumed item: 1

Consumer has consumed item: 3

Producer has produced item: 2
ripunjaynarula@LAPTOP-MOTVC22V:~$
```

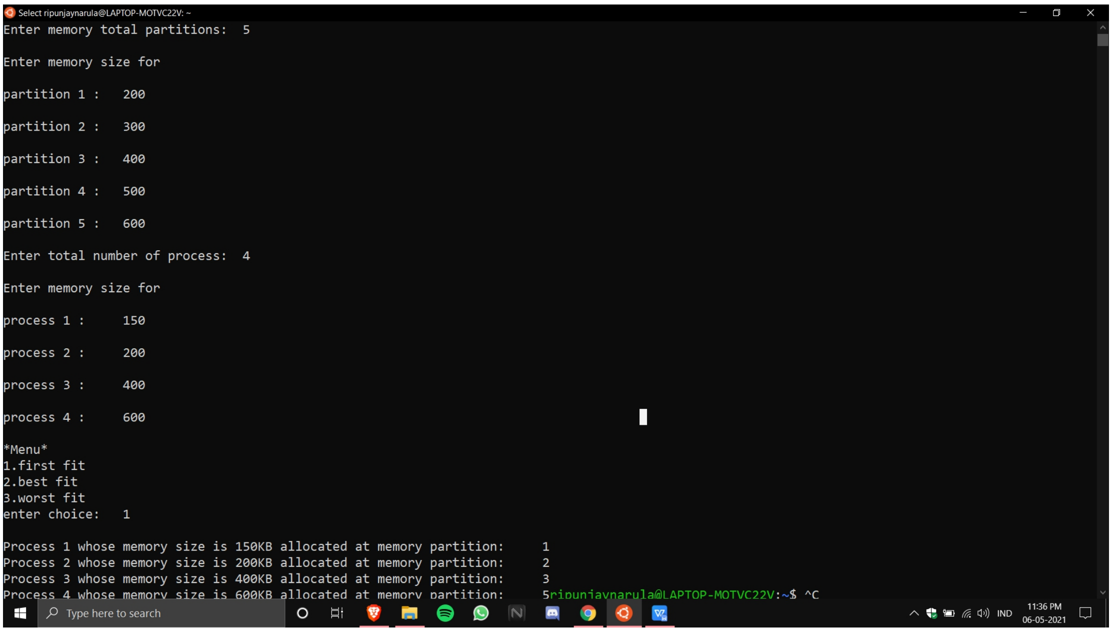
ASSESMENT-4

a) Consider a memory hole of size 1kb initially. When a sequence of memory request arrives as following, illustrate the memory allocation by various approaches and calculate the total amount memory wasted by external fragmentation and internal fragmentation in each approach.

```
#include <iostream>
using namespace std;
int main()
{
 int c,i,j,k,n,l,m[10],p[10],po[20],flag,z,y,temp,temp1;
    cout<<"Enter memory total partitions:\t";
    cin>>n;
    cout<<"\nEnter memory size for\n";
    for(i=1;i<=n;i++)
    {
     cout<<"\npartition "<<i<<" :\t";
     cin>>m[i];
     po[i]=i;
    }
    cout<<"\nEnter total number of process:\t";
    cin>>j;
    cout<<"\nEnter memory size for\n";
    for(i=1;i<=j;i++)
    {
    cout<<"\nprocess "<<i<<" :\t";
     cin>>p[i];
    }
    cout<<"\n**Menu**\n1.first fit\n2.best fit\n3.worst fit\nenter
choice:\t";
    cin>>c;
    switch(c)
    {
    case 1:
       for(i=1;i<=j;i++)
    {
      flag=1;
      for(k=1;k<=n;k++)
    {
      if(p[i]<=m[k])
      {
        cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
allocated at memory partition:\t"<<po[k];
        m[k]=m[k]-p[i];
        break;
      }
      else
      {
       flag++;
      }
    }
    if(flag>n)
    {
        cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
can't be allocated";
      }
      }
      break;
      case 2:
       for(y=1;y<=n;y++)
         {
         for(z=y;z<=n;z++)
         {
         if(m[y]>m[z])
         {
         temp=m[y];
         m[y]=m[z];
         m[z]=temp;
         temp1=po[y];
         po[y]=po[z];
         po[z]=temp1;
         }
         }
         }
         for(i=1;i<=j;i++)
      {
         flag=1;
         for(k=1;k<=n;k++)
      {
         if(p[i]<=m[k])
         {
             cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB allocated at memory partition:\t"<<po[k];
             m[k]=m[k]-p[i];
             break;
          }
          else
          {
            flag++;
          }
      }
      if(flag>n)
      {
         cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB can't be allocated";
      }
      }
          break;
          case 3:
          for(y=1;y<=n;y++)
          {
          for(z=y;z<=n;z++)
          {
          if(m[y]<m[z])
          {
          temp=m[y];
          m[y]=m[z];
          m[z]=temp;
          temp1=po[y];
          po[y]=po[z];
          po[z]=temp1;
          }
          }
          }
          for(i=1;i<=j;i++)
      {
          flag=1;
          for(k=1;k<=n;k++)
      {
          if(p[i]<=m[k])
          {
             cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
    allocated at memory partition:\t"<<po[k];
             m[k]=m[k]-p[i];
             break;
          }
          else
          {
            flag++;
          }
      }
      if(flag>n)
      {
         cout<<"\nProcess "<<i<<" whose memory size is "<<p[i]<<"KB
    can't be allocated";
      }
      }
```

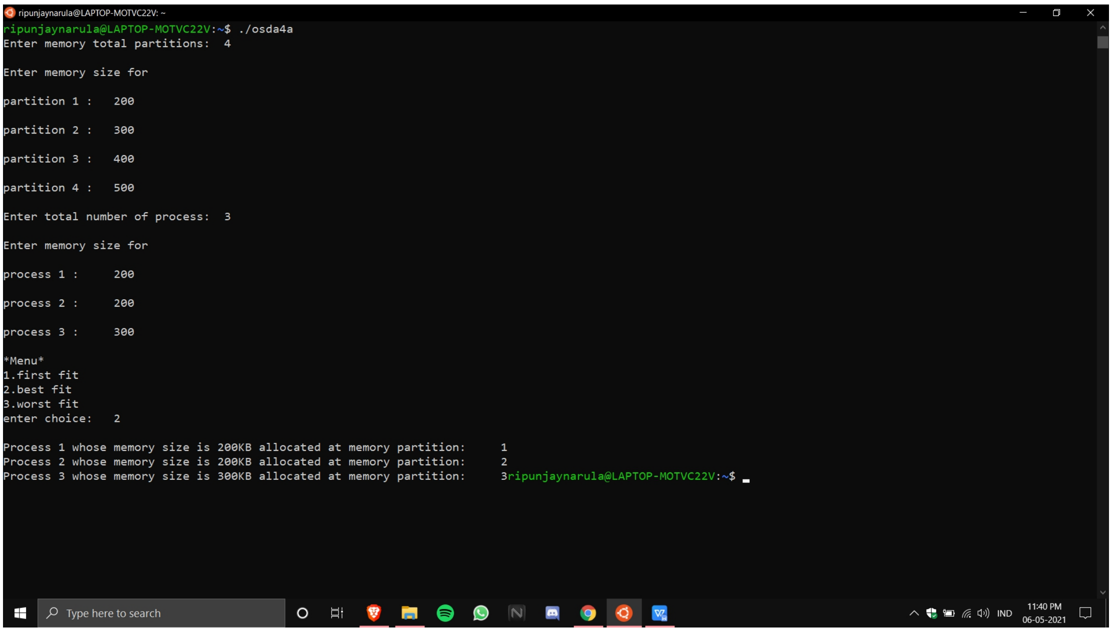
```
            break;
            }
        return 0;
}
```

i. First fit



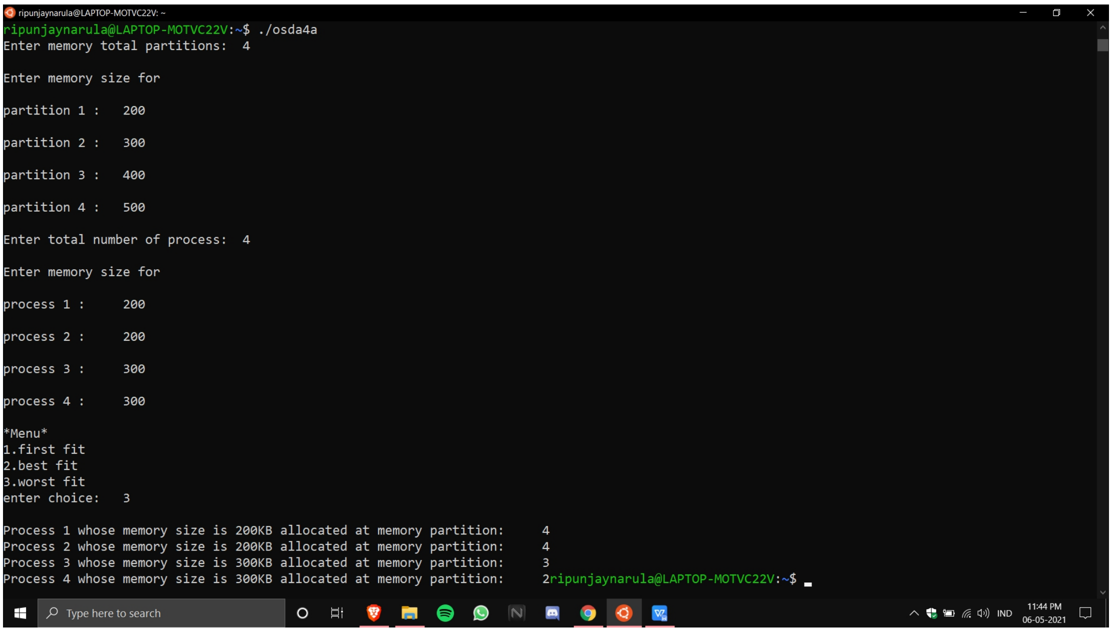
```
Enter memory total partitions:  5

Enter memory size for

partition 1 :   200

partition 2 :   300

partition 3 :   400

partition 4 :   500

partition 5 :   600

Enter total number of process:  4

Enter memory size for

process 1 :     150

process 2 :     200

process 3 :     400

process 4 :     600

*Menu*
1.first fit
2.best fit
3.worst fit
enter choice:   1

Process 1 whose memory size is 150KB allocated at memory partition:     1
Process 2 whose memory size is 200KB allocated at memory partition:     2
Process 3 whose memory size is 400KB allocated at memory partition:     3
Process 4 whose memory size is 600KB allocated at memory partition:     5ripunjaynarula@LAPTOP-MOTVC22V:~$ ^C
```

ii. Best fit



```
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./osda4a
Enter memory total partitions:  4

Enter memory size for

partition 1 :   200

partition 2 :   300

partition 3 :   400

partition 4 :   500

Enter total number of process:  3

Enter memory size for

process 1 :     200

process 2 :     200

process 3 :     300

*Menu*
1.first fit
2.best fit
3.worst fit
enter choice:   2

Process 1 whose memory size is 200KB allocated at memory partition:     1
Process 2 whose memory size is 200KB allocated at memory partition:     2
Process 3 whose memory size is 300KB allocated at memory partition:     3ripunjaynarula@LAPTOP-MOTVC22V:~$ _
```

iii. Worst fit



```
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./osda4a
Enter memory total partitions:  4

Enter memory size for

partition 1 :   200

partition 2 :   300

partition 3 :   400

partition 4 :   500

Enter total number of process:  4

Enter memory size for

process 1 :     200

process 2 :     200

process 3 :     300

process 4 :     300

*Menu*
1.first fit
2.best fit
3.worst fit
enter choice:   3

Process 1 whose memory size is 200KB allocated at memory partition:     4
Process 2 whose memory size is 200KB allocated at memory partition:     4
Process 3 whose memory size is 300KB allocated at memory partition:     3
Process 4 whose memory size is 300KB allocated at memory partition:     2ripunjaynarula@LAPTOP-MOTVC22V:~$ _
```

b) Write a program to implement the page replacement algorithms.

i. FIFO ii. LRU iii. OPT

```
#include<stdio.h>
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;

void getData()
{
    printf("\nEnter length of page reference sequence:");
    scanf("%d",&n);
    printf("\nEnter the page reference sequence:");
    for(i=0; i<n; i++)
        scanf("%d",&in[i]);
    printf("\nEnter no of frames:");
    scanf("%d",&nf);
}

void initialize()
{
    pgfaultcnt=0;
    for(i=0; i<nf; i++)
        p[i]=9999;
}
```

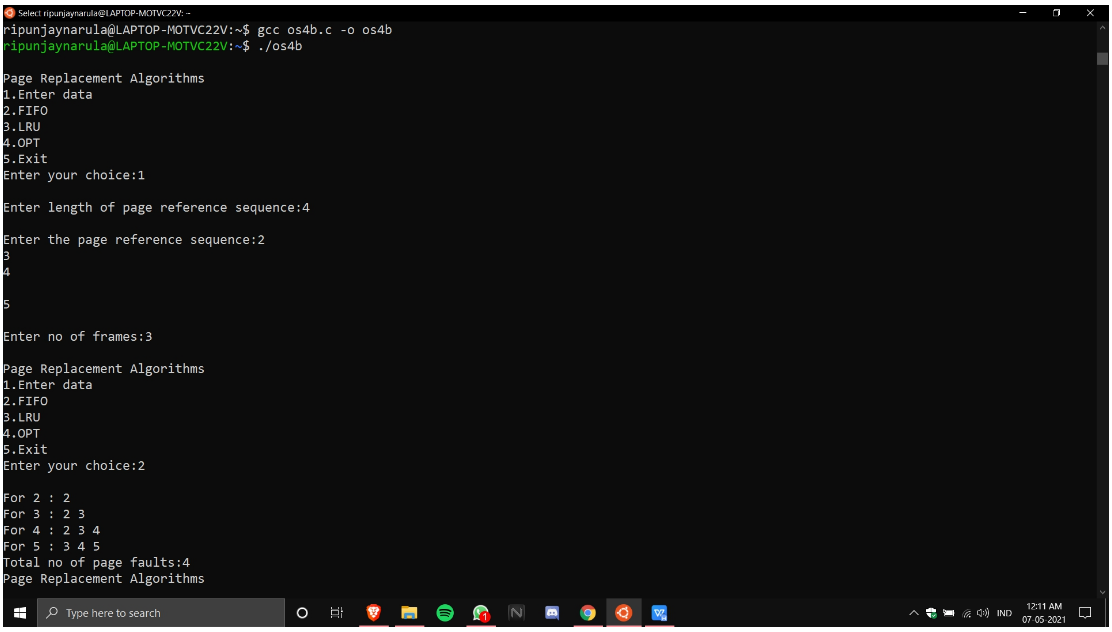
```
int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }

    }

    return hit;
}

int getHitIndex(int data)
{
    int hitind;
    for(k=0; k<nf; k++)
    {
        if(p[k]==data)
        {
            hitind=k;
            break;
        }
    }
    return hitind;
}

void dispPages()
{
    for (k=0; k<nf; k++)
    {
        if(p[k]!=9999)
            printf(" %d",p[k]);
    }

}

void dispPgFaultCnt()
{
    printf("\nTotal no of page faults:%d",pgfaultcnt);
}

void fifo()
{
    initialize();
    for(i=0; i<n; i++)
    {
        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {

            for(k=0; k<nf-1; k++)
                p[k]=p[k+1];

            p[k]=in[i];
            pgfaultcnt++;
            dispPages();
        }
        else
            printf("No page fault");
    }
    dispPgFaultCnt();
}


void optimal()
{
    initialize();
    int near[50];
    for(i=0; i<n; i++)
    {

        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {
```

```
for(j=0; j<nf; j++)
{
    int pg=p[j];
    int found=0;
    for(k=i; k<n; k++)
    {
        if(pg==in[k])
        {
            near[j]=k;
            found=1;
            break;
        }
        else
            found=0;
    }
    if(!found)
        near[j]=9999;
}
int max=-9999;
int repindex;
for(j=0; j<nf; j++)
{
    if(near[j]>max)
    {
        max=near[j];
        repindex=j;
```

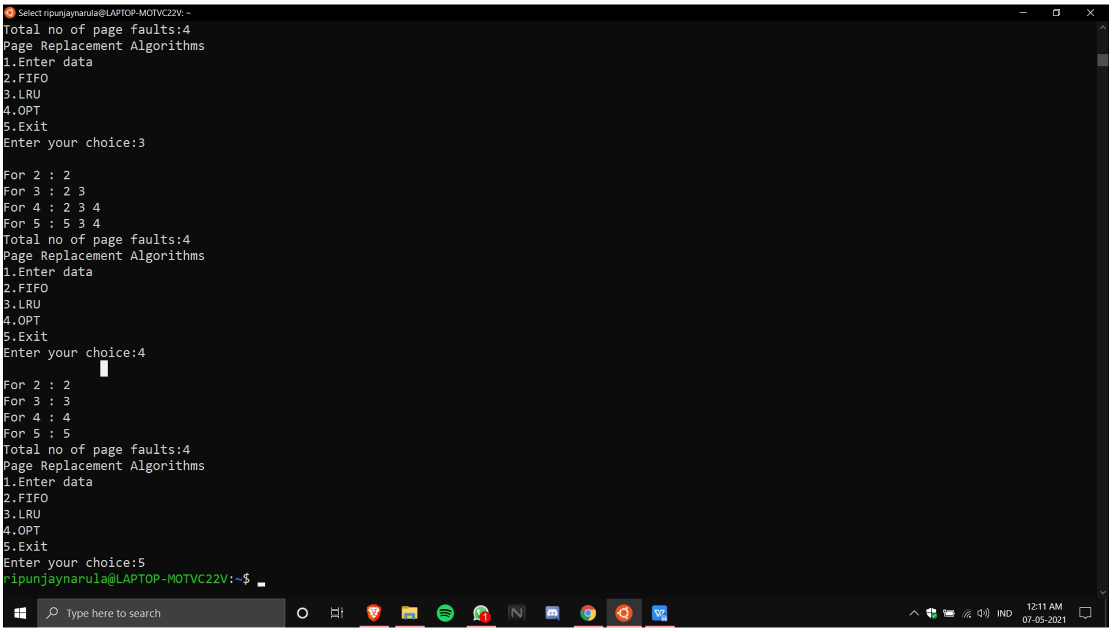
```
                }
            }
            p[repindex]=in[i];
            pgfaultcnt++;

            dispPages();
        }
        else
            printf("No page fault");
    }
    dispPgFaultCnt();
}

void lru()
{
    initialize();

    int least[50];
    for(i=0; i<n; i++)
    {

        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {

            for(j=0; j<nf; j++)
```

```
{
    int pg=p[j];
    int found=0;
    for(k=i-1; k>=0; k--)
    {
        if(pg==in[k])
        {
            least[j]=k;
            found=1;
            break;
        }
        else
            found=0;
    }
    if(!found)
        least[j]=-9999;
}
int min=9999;
int repindex;
for(j=0; j<nf; j++)
{
    if(least[j]<min)
    {
        min=least[j];
        repindex=j;
    }
}
```

```
            p[repindex]=in[i];
            pgfaultcnt++;

            dispPages();
        }
        else
            printf("No page fault!");
    }
    dispPgFaultCnt();
}

int main()
{
    int choice;
    while(1)
    {
        printf("\nPage Replacement Algorithms\n1.Enter
data\n2.FIFO\n3.LRU\n4.OPT\n5.Exit\nEnter your choice:");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            getData();
            break;
        case 2:
            fifo();
            break;
        case 3:
               lru();
               break;
          case 4:
optimal();
               break;
          default:
               return 0;
               break;
          }
     }
}
```



```
ripunjaynarula@LAPTOP-MOTVC22V:~$ gcc os4b.c -o os4b
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./os4b

Page Replacement Algorithms
1.Enter data
2.FIFO
3.LRU
4.OPT
5.Exit
Enter your choice:1

Enter length of page reference sequence:4

Enter the page reference sequence:2
3
4

5

Enter no of frames:3

Page Replacement Algorithms
1.Enter data
2.FIFO
3.LRU
4.OPT
5.Exit
Enter your choice:2

For 2 : 2
For 3 : 2 3
For 4 : 2 3 4
For 5 : 3 4 5
Total no of page faults:4
Page Replacement Algorithms
```



```
Total no of page faults:4
Page Replacement Algorithms
1.Enter data
2.FIFO
3.LRU
4.OPT
5.Exit
Enter your choice:3

For 2 : 2
For 3 : 2 3
For 4 : 2 3 4
For 5 : 5 3 4
Total no of page faults:4
Page Replacement Algorithms
1.Enter data
2.FIFO
3.LRU
4.OPT
5.Exit
Enter your choice:4

For 2 : 2
For 3 : 3
For 4 : 4
For 5 : 5
Total no of page faults:4
Page Replacement Algorithms
1.Enter data
2.FIFO
3.LRU
4.OPT
5.Exit
Enter your choice:5
ripunjaynarula@LAPTOP-MOTVC22V:~$
```

C) Write a program that implements the FIFO, LRU, and optimal pager replacement algorithms. First, generate a random page-reference string where page numbers range from 0 to 9. Apply the random page reference string to each algorithm, and record the number of page faults incurred by each algorithm. Implement the replacement algorithms so that the number of page frames can vary from 1 to 7. Assume that demand paging is used.

```
#include<stdio.h>
void FIFO();
void LRU();
void OPTIMAL();

int main()
{
    int ch;
    do
    {
        printf("\n\n1.FIFO\n2.LRU\n3.Optimal\n4.Exit\nEnter Choice : ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
            FIFO();
            break;
            case 2:
            LRU();
            break;
            case 3:
            OPTIMAL();
            break;
        }
    }while(ch!=4);
}
void FIFO()
{
    int frame[3]={-1,-1,-1},refer[20],ctn=0,i,j,number,flag;
    float ratio,hitctn=0.00;
    printf("\nEnter length of refererence string: ");
    scanf("%d",&number);
    printf("\nEnter refererence String with giving space:\n");
    for(i=0;i<number;i++)
    scanf("%d",&refer[i]);
    for(i=0;i<number;i++)
    {
        flag=0;
        for(j=0;j<3;j++)
        if(frame[j]==refer[i])
        {
           printf("\nPage Hit ");
           hitctn++;
           flag=1;
           break;
        }

        if(flag==0)
        {
           printf("\nPage Miss");
           printf(" Before:");
           for(j=0;j<3;j++)
           printf(" %d",frame[j]);
           frame[ctn]=refer[i];
           ctn++;
           if(ctn>=3)
           ctn=0;
           printf(" After:");
           for(j=0;j<3;j++)
           printf(" %d",frame[j]);
        }
    }
ratio=hitctn/number;
printf("\n\nHit ratio = %f ",ratio);
}
```

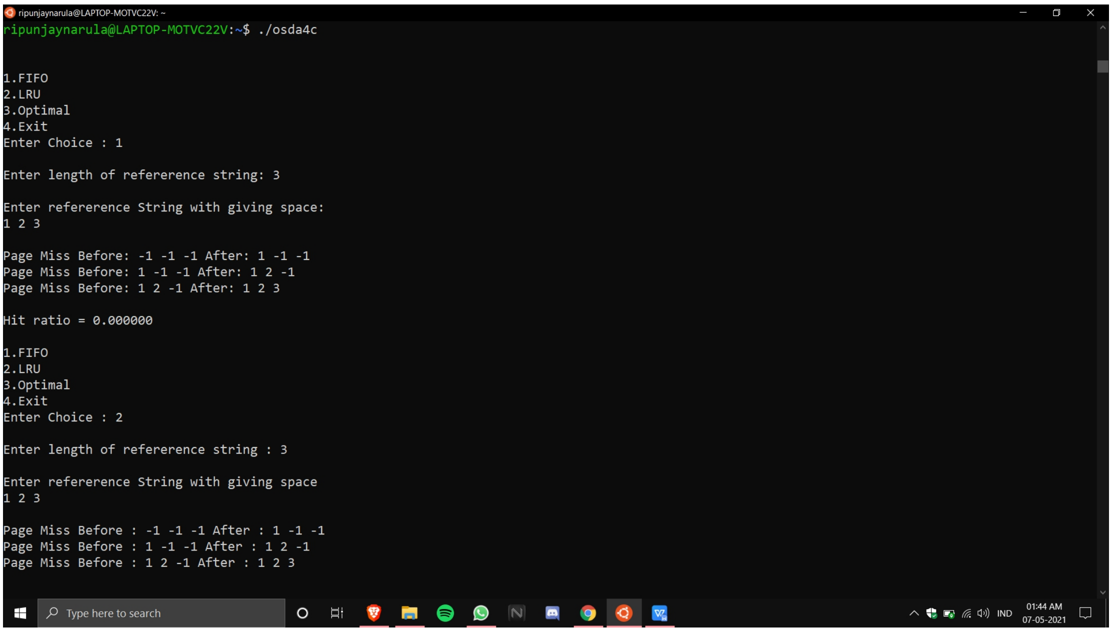
```
void LRU()
{
    int frame[3]={-1,-1,-1},used[3]={-1,-1,-
1},ctn=0,refer[20],i,j,flag,number,index,value;
    float ratio,hitctn=0;
    printf("\nEnter length of refererence string : ");
    scanf("%d",&number);
    printf("\nEnter refererence String with giving space \n");
    for(i=0;i<number;i++)
    scanf("%d",&refer[i]);
    for(i=0;i<number;i++)
    {
        flag=0;
        for(j=0;j<3;j++)
        if(frame[j]==refer[i])
        {
            printf("\nPage Hit ");
            hitctn++;
            flag=1;
            used[j]=ctn;
            break;
        }
        if(flag==0)
        {
            printf("\nPage Miss");
            printf(" Before :");
            for(j=0;j<3;j++)
            printf(" %d",frame[j]);
            index=0;
            value=used[0];
            if(ctn!=0) {
            for(j=0;j<3;j++)
            if(value>used[j]&&value!=used[j])
            {
                index=j;
                value=used[j];
            }
        }
        frame[index]=refer[i];
        used[index]=ctn;
        printf(" After :");
        for(j=0;j<3;j++)
        printf(" %d",frame[j]);
    }
ctn++;
}
ratio=hitctn/number;
printf("\n\nHit ratio = %f ",ratio);
}
void OPTIMAL()
{
    int frame[3]={-1,-1,-1},used[3]={-1,-1,-
1},ctn=0,refer[20],i,j,flag,number,value1,value2,value3,index;
    float ratio,hitctn=0;
    printf("\nEnter length of refererence string : ");
    scanf("%d",&number);
```

```
printf("\nEnter refererence String with giving space \n");
for(i=0;i<number;i++)
scanf("%d",&refer[i]);
for(i=0;i<number;i++)
{
   flag=0;
   for(j=0;j<3;j++)
   if(frame[j]==refer[i])
   {
      flag=1;
      printf("\nPage Hit");
      hitctn++;
      break;
   }
if(flag==0)
{
printf("\nPage Miss");
if(ctn<3)
{
   frame[ctn]=refer[i];
   printf("\tStatus :");
   for(j=0;j<3;j++)
   printf(" %d",frame[j]);
   ctn++;
}
else
{
```

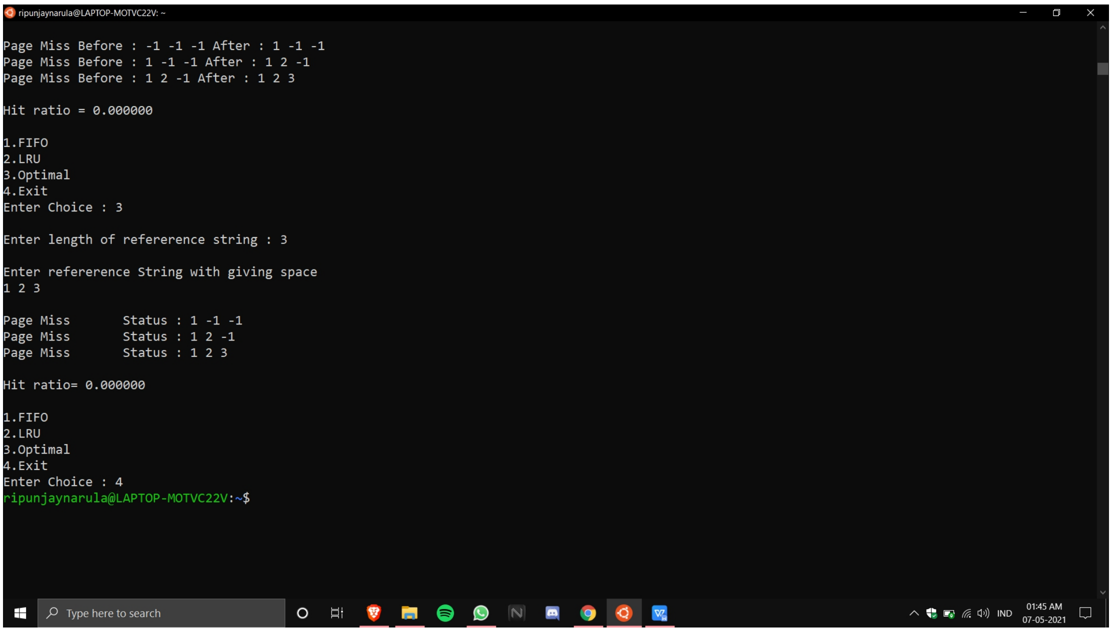
```
printf(" Before :");
for(j=0;j<3;j++)
printf(" %d",frame[j]);
value1=frame[0];
flag=0;
for(j=i;j<number;j++)
if(refer[j]==value1)
{
   value1=j;
   flag=1;
   break;
}
if(flag==0)
value1=number;
value2=frame[1];
flag=0;
for(j=i;j<number;j++)
if(refer[j]==value2)
{
   value2=j;
   flag=1;
   break;
}
if(flag==0)
value2=number;
value3=frame[2];
flag=0;
```

```
        for(j=i;j<number;j++)
        if(refer[j]==value3)
        {
            value3=j;
            flag=1;
            break;
        }
        if(flag==0)
            value3=number;
        if(value1<value2)
        if(value3<value2)
            index=1;
        else
            index=2;
        else
        if(value3<value1)
            index=0;
        else
            index=2;


        frame[index]=refer[i];
        printf(" After:");
        for(j=0;j<3;j++)
            printf(" %d",frame[j]);
        }
    }
}
```

```
ratio=hitctn/number;
printf("\n\nHit ratio= %f ",ratio);
}
```



```
ripunjaynarula@LAPTOP-MOTVC22V:~$ ./osda4c


1.FIFO
2.LRU
3.Optimal
4.Exit
Enter Choice : 1

Enter length of refererence string: 3

Enter refererence String with giving space:
1 2 3

Page Miss Before: -1 -1 -1 After: 1 -1 -1
Page Miss Before: 1 -1 -1 After: 1 2 -1
Page Miss Before: 1 2 -1 After: 1 2 3

Hit ratio = 0.000000

1.FIFO
2.LRU
3.Optimal
4.Exit
Enter Choice : 2

Enter length of refererence string : 3

Enter refererence String with giving space
1 2 3

Page Miss Before : -1 -1 -1 After : 1 -1 -1
Page Miss Before : 1 -1 -1 After : 1 2 -1
Page Miss Before : 1 2 -1 After : 1 2 3
```



```
Page Miss Before : -1 -1 -1 After : 1 -1 -1
Page Miss Before : 1 -1 -1 After : 1 2 -1
Page Miss Before : 1 2 -1 After : 1 2 3

Hit ratio = 0.000000

1.FIFO
2.LRU
3.Optimal
4.Exit
Enter Choice : 3

Enter length of refererence string : 3

Enter refererence String with giving space
1 2 3

Page Miss       Status : 1 -1 -1
Page Miss       Status : 1 2 -1
Page Miss       Status : 1 2 3

Hit ratio= 0.000000

1.FIFO
2.LRU
3.Optimal
4.Exit
Enter Choice : 4
ripunjaynarula@LAPTOP-MOTVC22V:~$
```