OPERATING SYSTEMS - COM301P

LAB ASSIGNMENT - 7

SYNCHRONIZATION

VIKNESH RAJARAMON

COE18B060

1) Simulate the Producer Consumer code discussed in the class.

LOGIC:

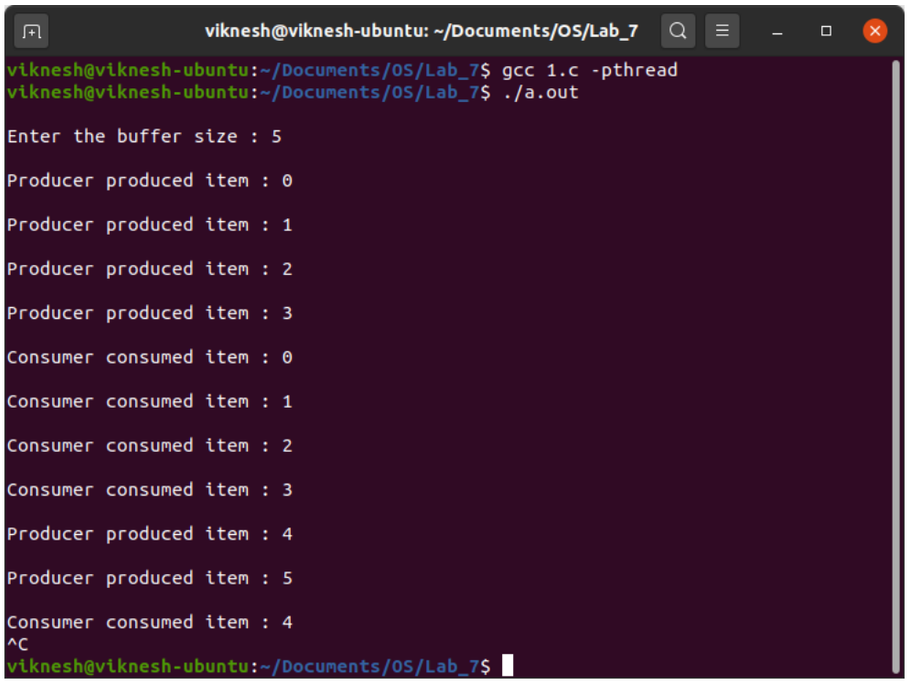
- We create two threads - **tid_producer** and **tid_consumer**.
- **tid_producer** runs the **producer code** while **tid_consumer** runs the **consumer code**.
- In **producer**, an item is **produced** only when the **Buffer is not full**.
- In **consumer**, an item is **consumed** only when the **Buffer is not empty**.
- The **buffer array** is implemented using a **circular array having two pointers** - **in** and **out**.
- **in pointer** points to the **next free position** in the buffer array.
- **out pointer** points to the **first full position** in the buffer array.

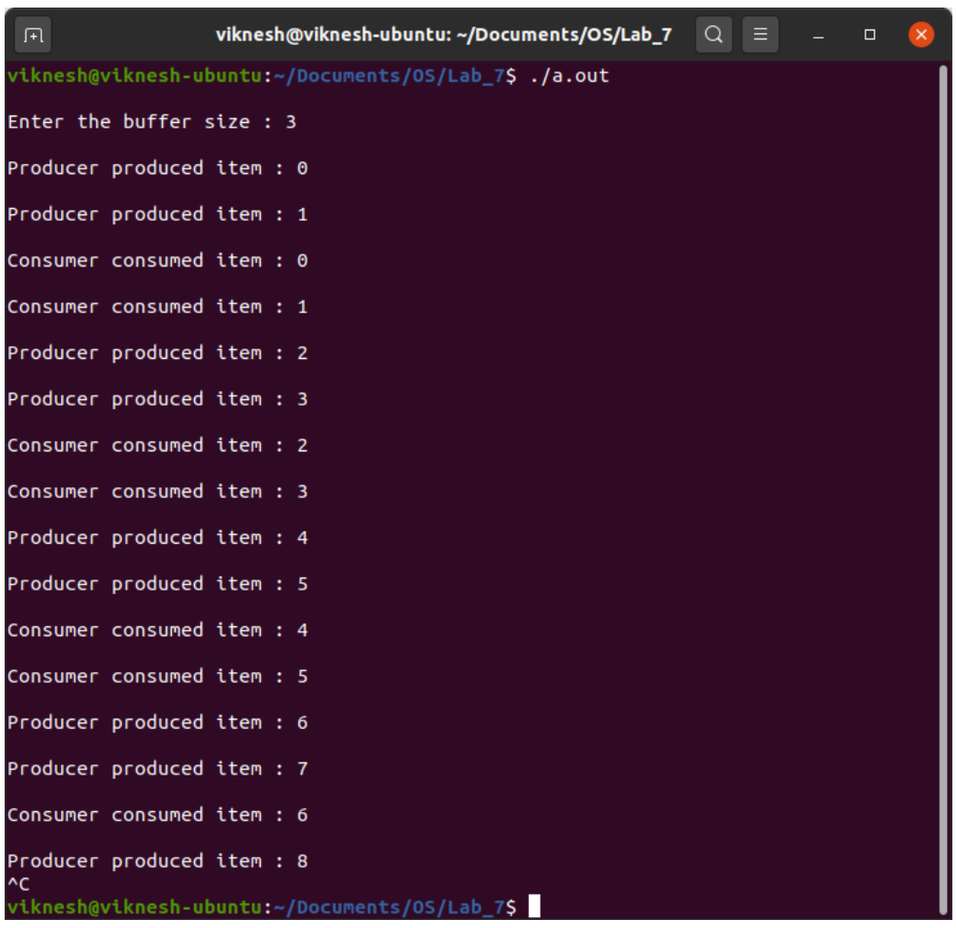
C CODE:

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

int buffer_size,in=0,out=0,items=0,total=0;

void *producer(void *param)
{
    int *buffer=param;
    while(1)
    {
            while(((in+1)%buffer_size)==out)
            {
                    ;
            }
            printf("\nProducer produced item : %d\n",total);
            buffer[in]=total;
            in=(in+1)%buffer_size;
            ++items;
            ++total;
    }
}

void *consumer(void *param)
{
    int *buffer=param;
    while(1)
    {
            while(in==out)
            {
                    ;
            }
            printf("\nConsumer consumed item : %d\n",buffer[out]);
            out=(out+1)%buffer_size;
            --items;
    }
}

int main()
{
    pthread_t tid_producer,tid_consumer;
    printf("\nEnter the buffer size : ");
    scanf("%d",&buffer_size);
```

```
    int buffer[buffer_size];
    pthread_create(&tid_producer,NULL,producer,buffer);
    pthread_create(&tid_consumer,NULL,consumer,buffer);
    pthread_join(tid_producer,NULL);
    pthread_join(tid_consumer,NULL);
    return 0;
}
```

OUTPUT:





2) Extend the producer consumer simulation in Q1 to sync access of critical data using Petersons

algorithm.

LOGIC:

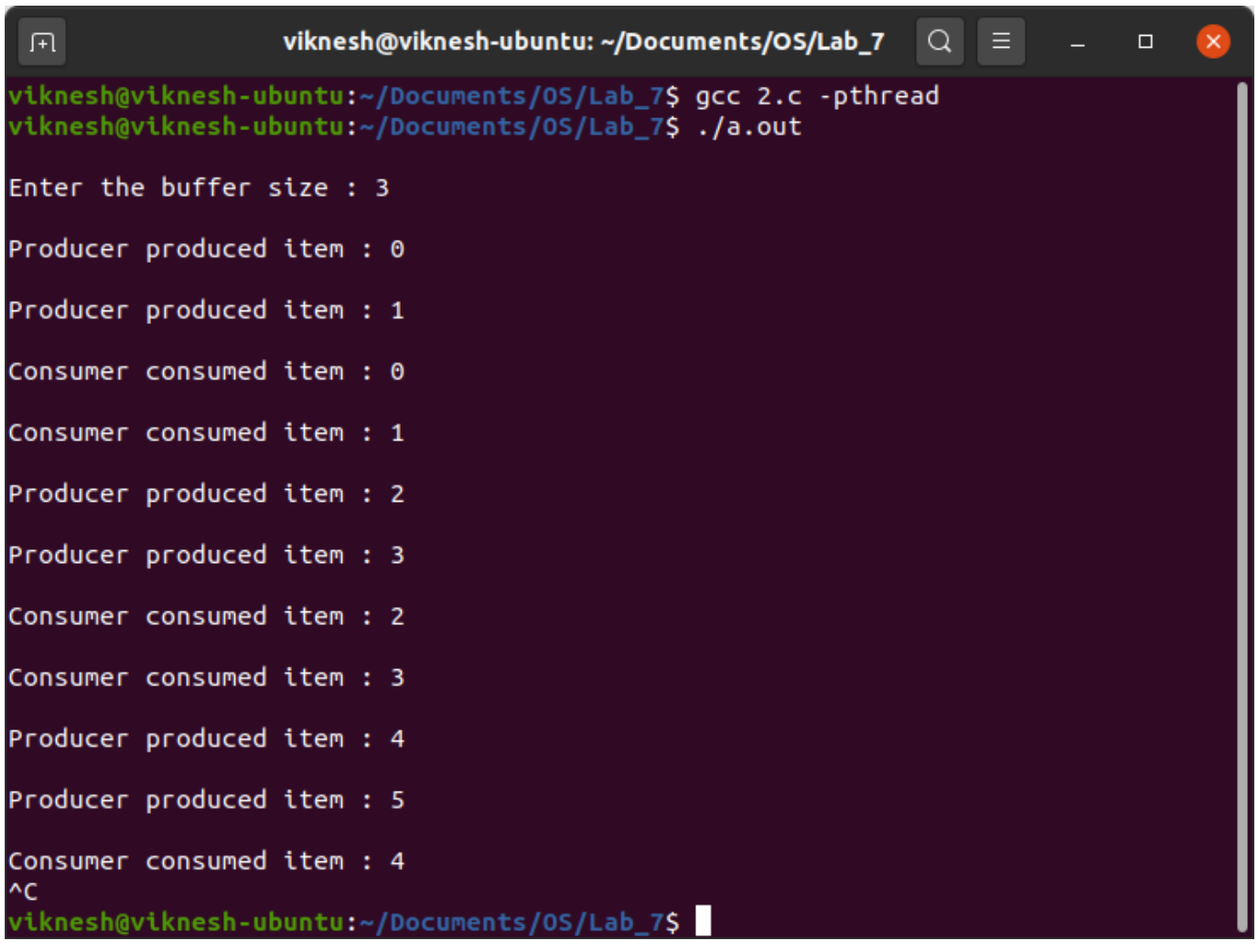
- We create two threads - **tid_producer** and **tid_consumer**.
- **tid_producer** runs the **producer code** while **tid_consumer** runs the **consumer code**.
- In **producer**, an item is **produced** only when the **Buffer is not full**.
- In **consumer**, an item is **consumed** only when the **Buffer is not empty**.
- The **buffer array** is implemented using a **circular array having two pointers** - **in** and **out**.
- **in pointer** points to the **next free position** in the buffer array.
- **out pointer** points to the **first full position** in the buffer array.

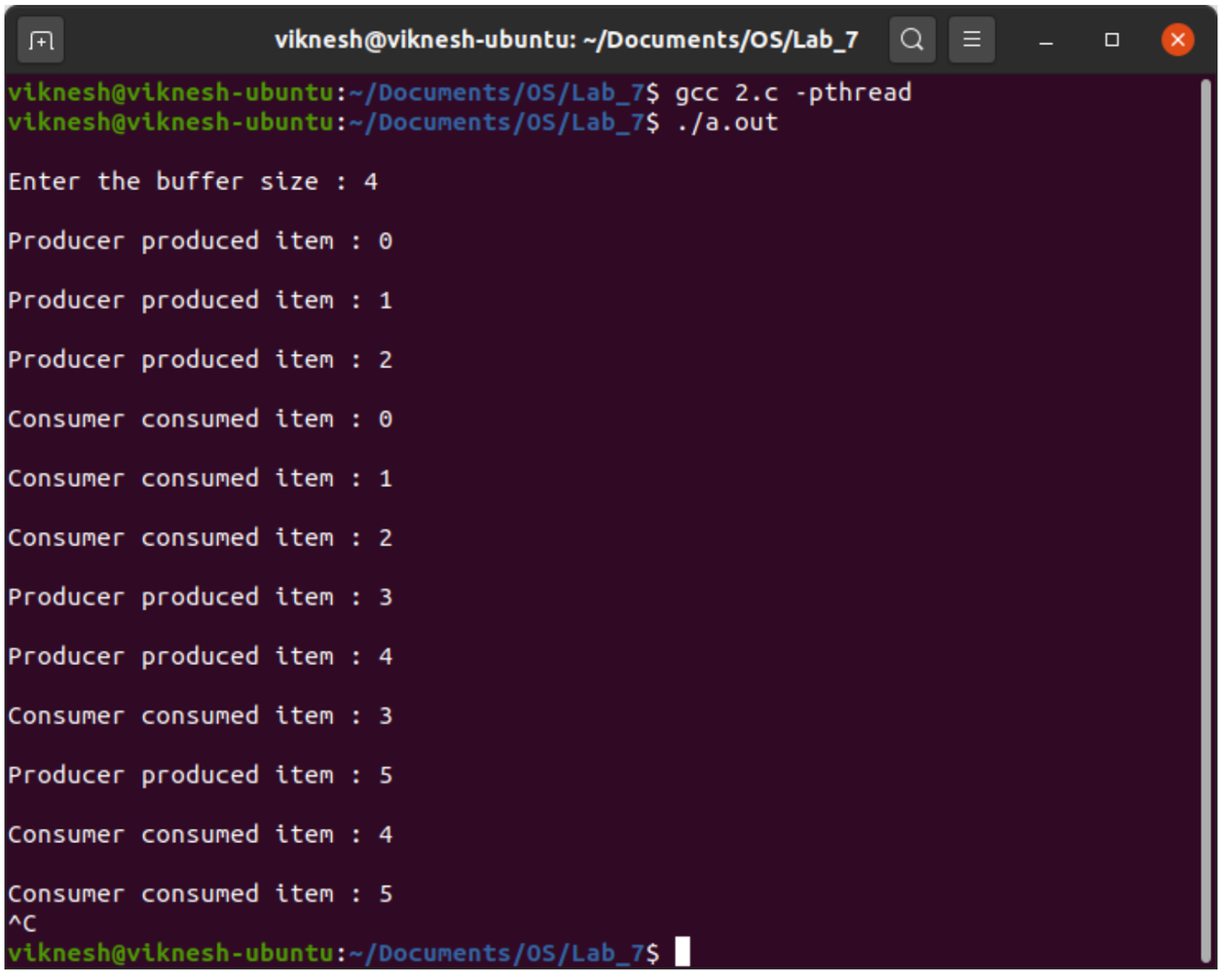
C CODE:

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

#define CONSUMER 0
#define PRODUCER 1

int buffer_size,in=0,out=0,items=0,total=0;
bool flag[2];
int turn;

void *producer(void *param)
{
    srand(time(0));
    int *buffer=param;
    int y=0;
    while(1)
    {
        flag[PRODUCER]=true;
        turn=CONSUMER;
        while((flag[CONSUMER]==true) && (turn==CONSUMER))
        {
            ;
        }
        y=rand()%buffer_size;
        for(int i=0;i<y;++i)
        {
            if(((in+1)%buffer_size)==out)
            {
                break;
            }
            else
            {
                printf("\nProducer produced item : %d\n",total);
                buffer[in]=total;
                in=(in+1)%buffer_size;
                ++items;
                ++total;
            }
        }
        flag[PRODUCER]=false;
```

```
    }
}

void *consumer(void *param)
{
    srand(time(0));
    int *buffer=param;
    int y=0;
    while(1)
    {
        flag[CONSUMER]=true;
        turn=PRODUCER;
        while((flag[PRODUCER]==true) && (turn==PRODUCER))
        {
            ;
        }
        y=rand()%buffer_size;
        for(int i=0;i<y;++i)
        {
            if(in==out)
            {
                break;
            }
            else
            {
                flag[PRODUCER]=false;
                printf("\nConsumer consumed item : %d\n",buffer[out]);
                out=(out+1)%buffer_size;
                --items;
            }
        }
        flag[CONSUMER]=false;
    }
}

int main()
{
    pthread_t tid_producer,tid_consumer;
    printf("\nEnter the buffer size : ");
    scanf("%d",&buffer_size);
    int buffer[buffer_size];
    pthread_create(&tid_producer,NULL,producer,buffer);
    pthread_create(&tid_consumer,NULL,consumer,buffer);
    pthread_join(tid_producer,NULL);
    pthread_join(tid_consumer,NULL);
    return 0;
}
```

OUTPUT:





3) **Dictionary Problem :** Let the producer set up a dictionary of at least 20 words with three

attributes (Word, Primary meaning Secondary meaning) and let the consumer search for the word and retrieve its respective primary and secondary meaning.

NOTE: This can be implemented using either Mutex locks or Petersons algorithm

LOGIC:

- We create two threads - **tid_producer** and **tid_consumer**.
- **tid_producer** runs the **producer code** while **tid_consumer** runs the **consumer code**.
- **Buffer size is unlimited** as we are using the **Binary Search tree** for storing the entries.
- In **consumer**, an item is **consumed** only when the **Buffer is not empty**.
- In **consumer code**, we **do not remove the dictionary entry from the dictionary**. We only search for the word and display it if the word exists in the dictionary.
- This has been done using Peterson's algorithm.

C CODE:

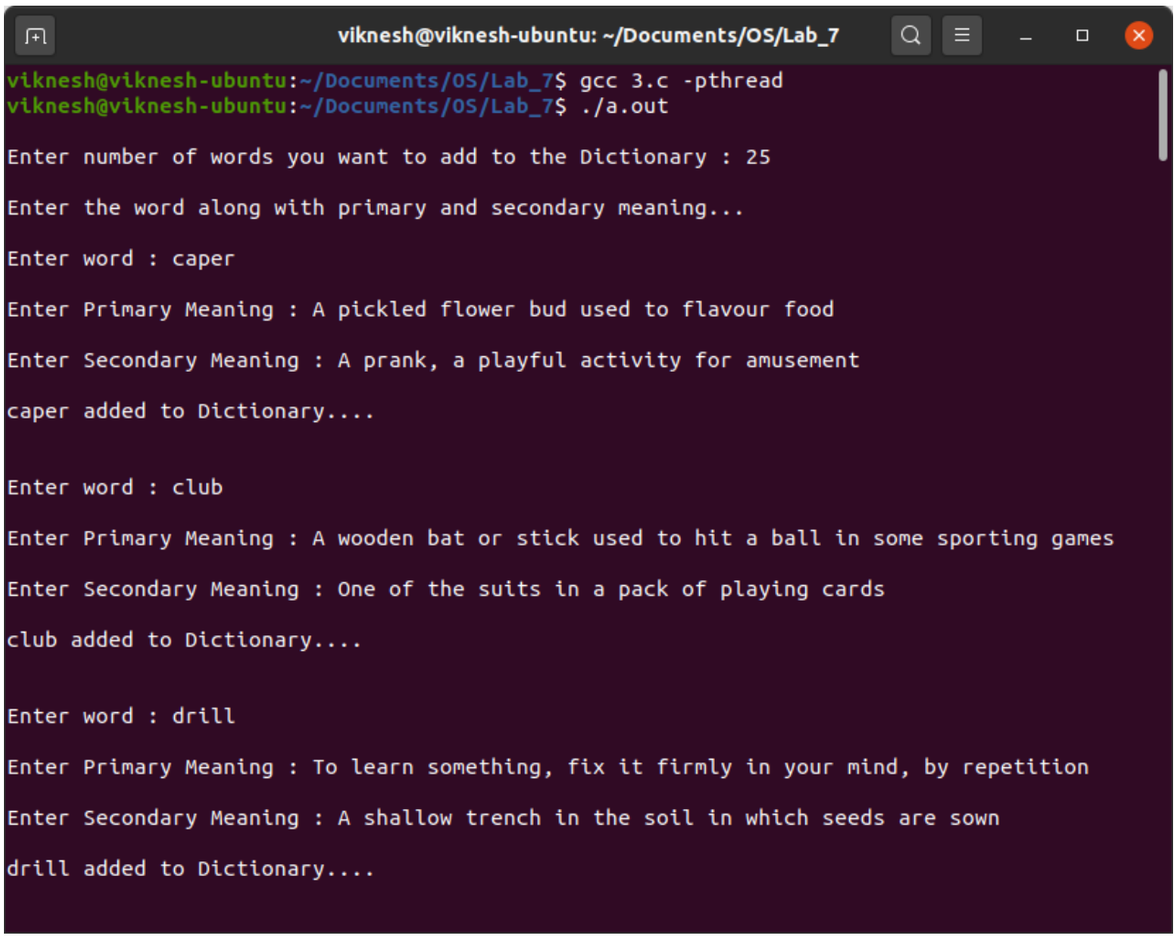
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#include<stdbool.h>
#include<pthread.h>

#define CONSUMER 0
#define PRODUCER 1

bool flag[2];
int turn;

int chance=0;

struct dict
{
    char word[100];
    char primary_meaning[1000];
    char secondary_meaning[1000];
    struct dict *left;
    struct dict *right;
};

struct dict *root=NULL;

int strcmpi(char s1[],char s2[])
{
    int i;
    if(strlen(s1)!=strlen(s2))
    {
        return -1;
    }
    for(i=0;i<strlen(s1);i++)
    {
        if(toupper(s1[i])!=toupper(s2[i]))
        {
```

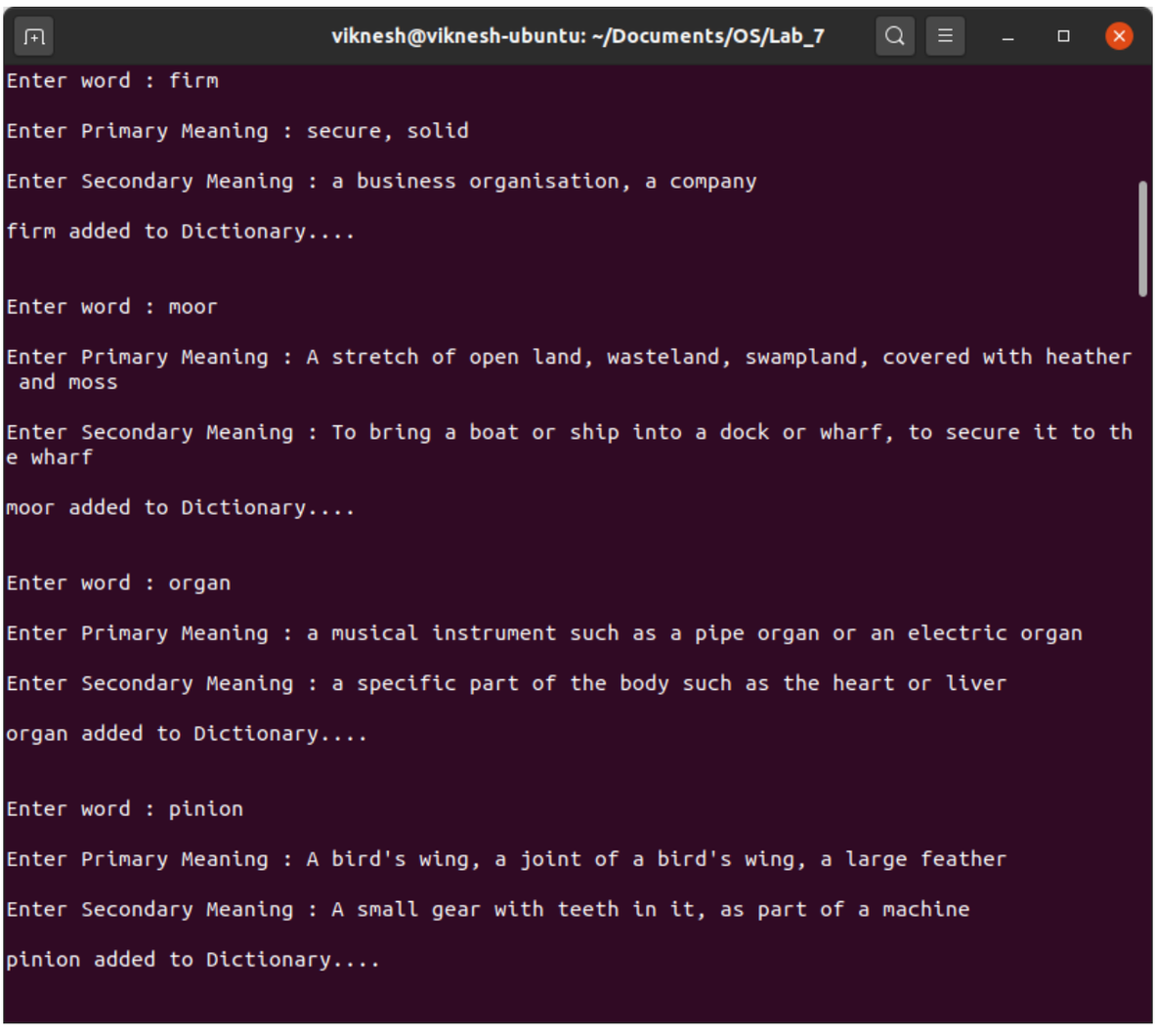
```
                return s1[i]-s2[i];
            }
        }
        return 0;
}

struct dict *insert(struct dict *ptr,struct dict *temp)
{
        if(ptr==NULL)
        {
                return temp;
        }
        else if(strcmpi(ptr->word,temp->word)>0)
        {
                ptr->left=insert(ptr->left,temp);
        }
        else
        {
                ptr->right=insert(ptr->right,temp);
        }
        return ptr;
}

void add_new_word()
{
        struct dict *temp=(struct dict *)malloc(sizeof(struct dict));
        temp->left=temp->right=NULL;
        getchar();
        printf("\nEnter word : ");
        scanf("%s",temp->word);
        getchar();
        printf("\nEnter Primary Meaning : ");
        fgets(temp->primary_meaning,sizeof(temp->primary_meaning),stdin);
        printf("\nEnter Secondary Meaning : ");
        fgets(temp->secondary_meaning,sizeof(temp->secondary_meaning),stdin);

        int len=strlen(temp->primary_meaning);
        temp->primary_meaning[len-1]=temp->primary_meaning[len];

        len=strlen(temp->secondary_meaning);
        temp->secondary_meaning[len-1]=temp->secondary_meaning[len];

        root=insert(root,temp);

        printf("\n%s added to Dictionary....\n",temp->word);
}

struct dict *search_word(char w[])
{
        struct dict *ptr=root;
        while(ptr!=NULL)
        {
                int diff=strcmpi(ptr->word,w);
                if(diff==0)
                {
                        break;
                }
                else if(diff>0)
                {
```

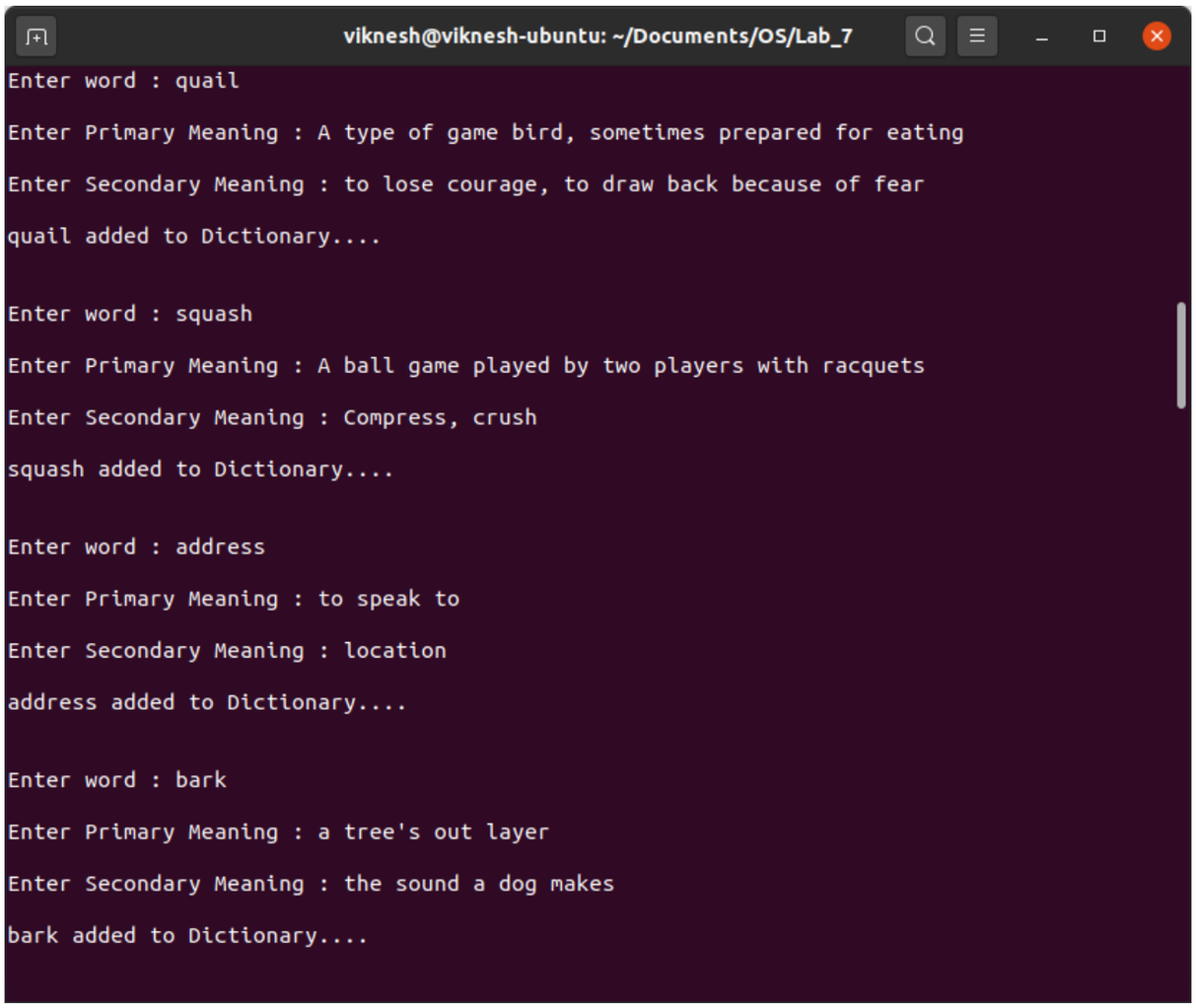
```
                ptr=ptr->left;
        }
        else
        {
                ptr=ptr->right;
        }
    }
    return ptr;
}

void print(struct dict *ptr)
{
    if(ptr==NULL)
    {
        return;
    }
    print(ptr->left);
    printf("%s, ",ptr->word);
    print(ptr->right);
}

void *producer()
{
    int y=0;
    while(1)
    {
        flag[PRODUCER]=true;
        turn=CONSUMER;
        while((flag[CONSUMER]==true) && (turn==CONSUMER))
        {
            ;
        }
        printf("\nEnter number of words you want to add to the Dictionary : ");
        scanf("%d",&y);
        printf("\nEnter the word along with primary and secondary meaning...\n");
        for(int i=0;i<y;++i)
        {
            add_new_word();
            chance=1;
        }
        printf("\n--------------------------------------------------------------------------------------------\n");
        flag[PRODUCER]=false;
    }
}

void *consumer()
{
    int y=0;
    char w[100];
    struct dict *ptr=NULL;
    while(1)
    {
        flag[CONSUMER]=true;
        turn=PRODUCER;
        while((flag[PRODUCER]==true) && (turn==PRODUCER))
        {
            ;
        }
        if(chance==0)
```

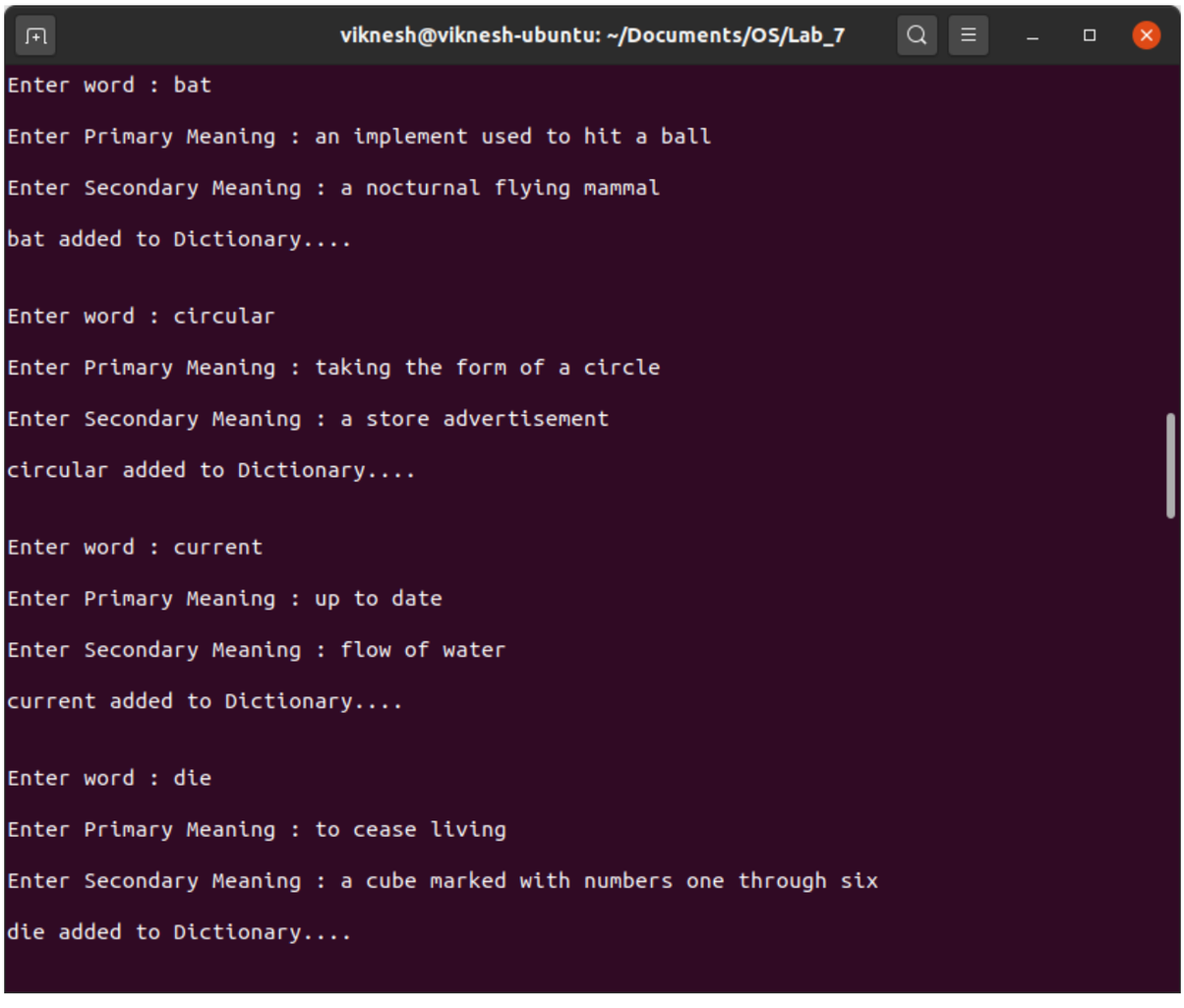
```
            {
                goto end;
            }
            printf("\nWords in Dictionary : ");
            print(root);
            printf("\b\b\n");
            printf("\nEnter number of words you want to search in the Dictionary : ");
            scanf("%d",&y);
            for(int i=0;i<y;++i)
            {
                if(root==NULL)
                {
                    printf("\nDictionary is empty...\n");
                    break;
                }
                else
                {
                    getchar();
                    printf("\nEnter the word to be searched for  : ");
                    scanf("%s",w);
                    ptr=search_word(w);
                    if(ptr!=NULL)
                    {
                        printf("\nPrimary Meaning : %s",ptr->primary_meaning);
                        printf("\nSecondary Meaning : %s\n",ptr->secondary_meaning);
                    }
                    else
                    {
                        printf("\nWord not found\n");
                    }
                }
            }
            printf("\n--------------------------------------------------------------------------------------------\n");
            end:
            flag[CONSUMER]=false;
    }
}

int main()
{
    pthread_t tid_producer,tid_consumer;
    pthread_create(&tid_producer,NULL,producer,NULL);
    pthread_create(&tid_consumer,NULL,consumer,NULL);
    pthread_join(tid_producer,NULL);
    pthread_join(tid_consumer,NULL);
    return 0;
}
```

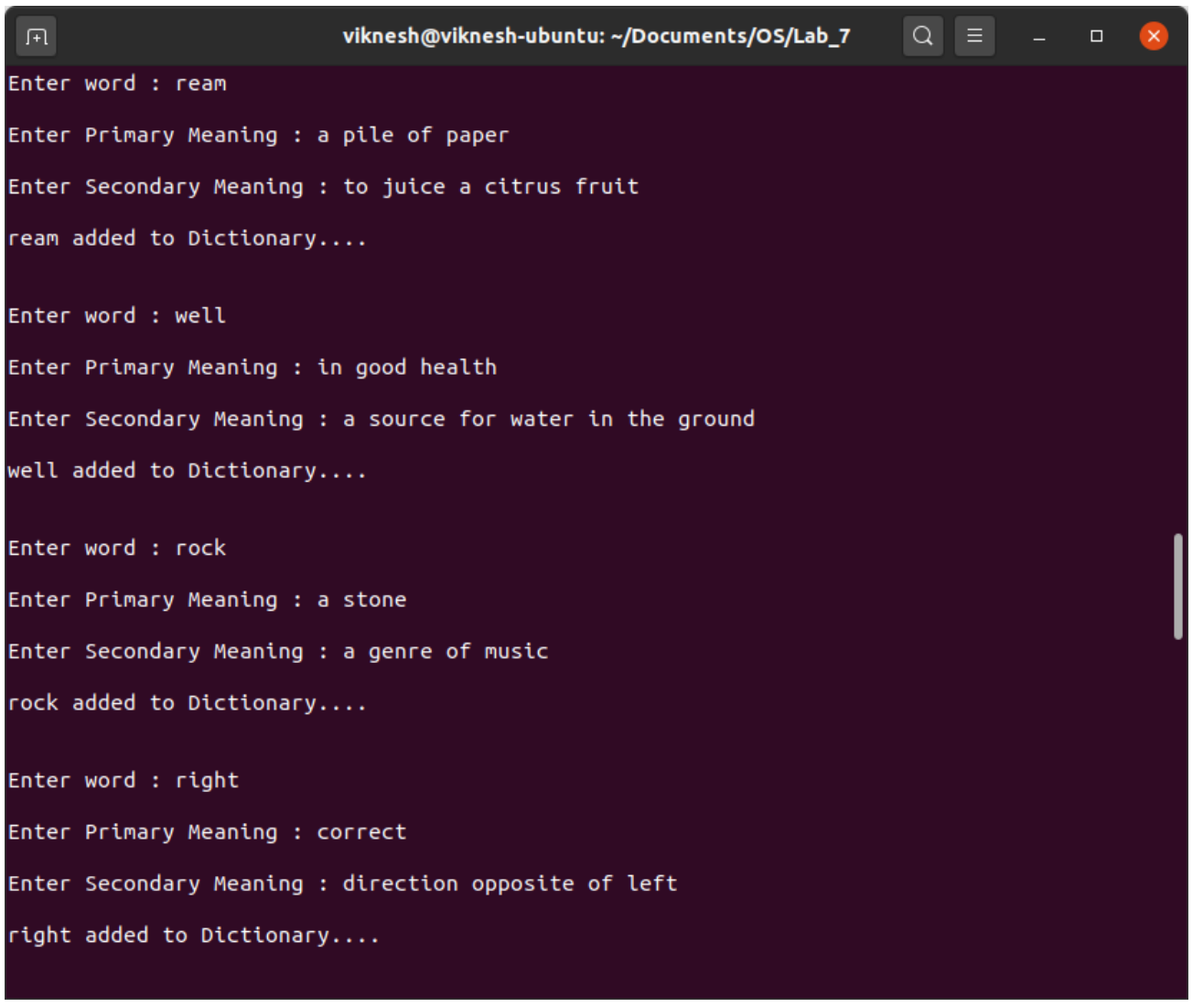
OUTPUT:

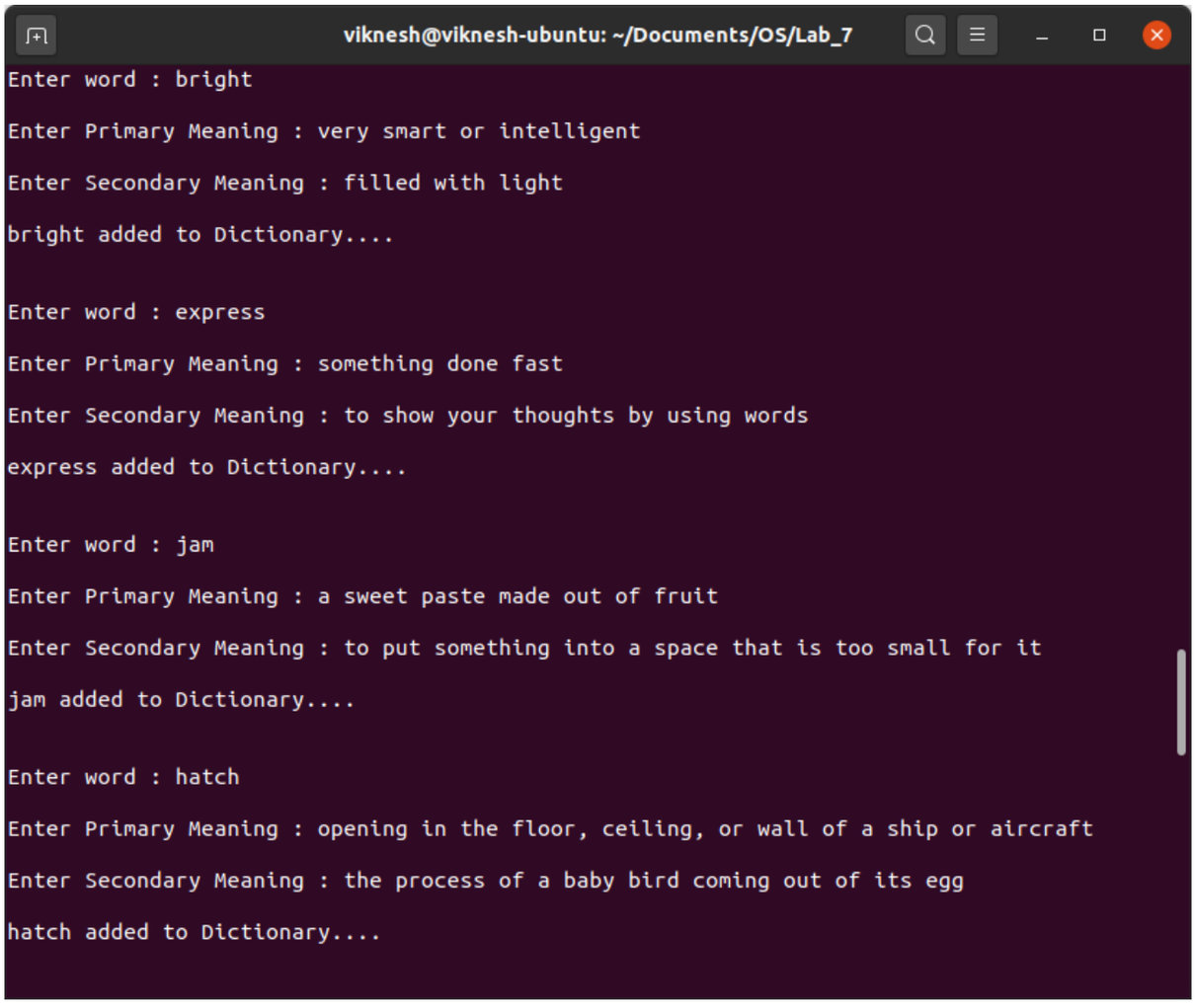

viknesh@viknesh-ubuntu: ~/Documents/OS/Lab_7
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$ gcc 3.c -pthread
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$ ./a.out
Enter number of words you want to add to the Dictionary : 25
Enter the word along with primary and secondary meaning...
Enter word : caper
Enter Primary Meaning : A pickled flower bud used to flavour food
Enter Secondary Meaning : A prank, a playful activity for amusement
caper added to Dictionary....
Enter word : club
Enter Primary Meaning : A wooden bat or stick used to hit a ball in some sporting games
Enter Secondary Meaning : One of the suits in a pack of playing cards
club added to Dictionary....
Enter word : drill
Enter Primary Meaning : To learn something, fix it firmly in your mind, by repetition
Enter Secondary Meaning : A shallow trench in the soil in which seeds are sown
drill added to Dictionary....

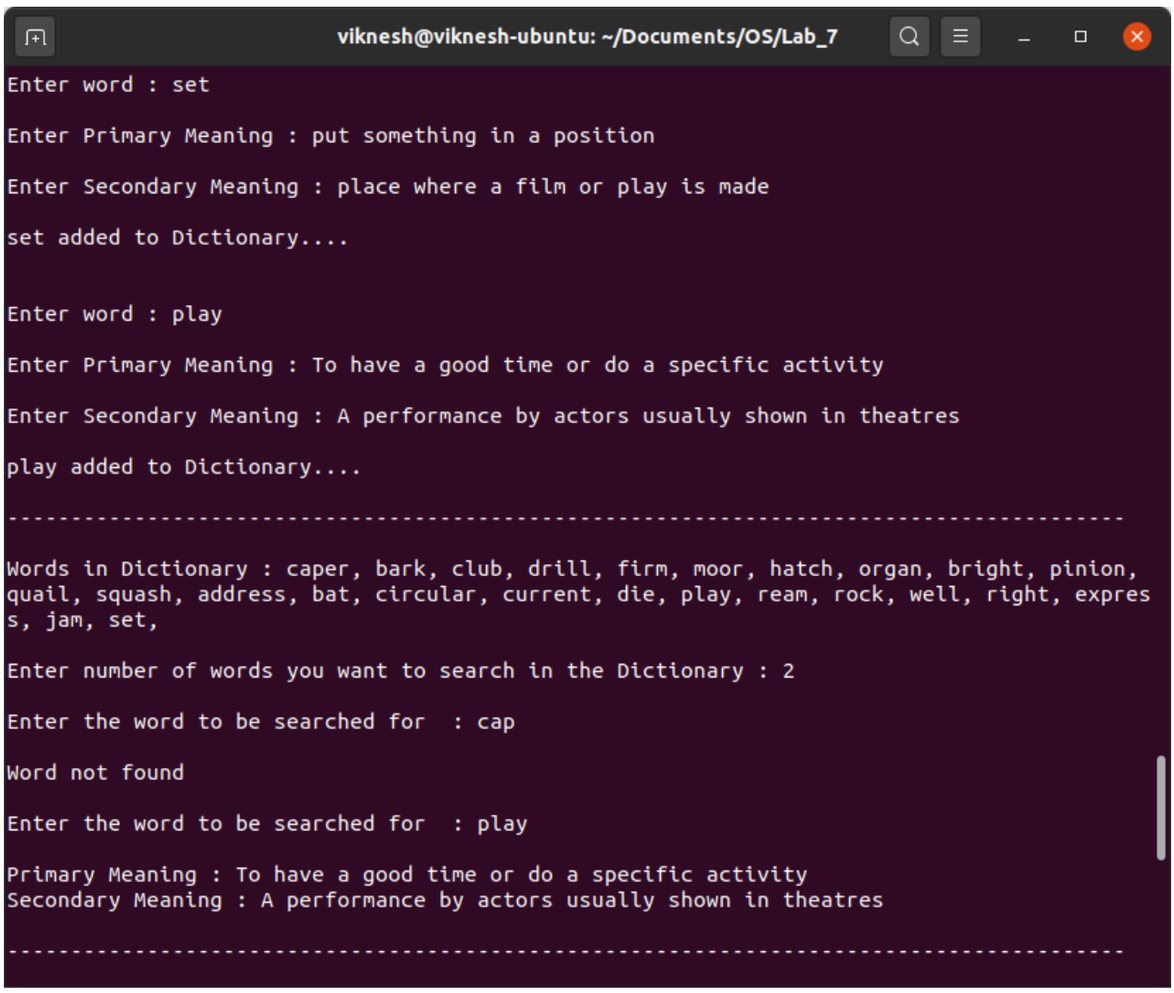

viknesh@viknesh-ubuntu: ~/Documents/OS/Lab_7
Enter word : firm
Enter Primary Meaning : secure, solid
Enter Secondary Meaning : a business organisation, a company
firm added to Dictionary....
Enter word : moor
Enter Primary Meaning : A stretch of open land, wasteland, swampland, covered with heather and moss
Enter Secondary Meaning : To bring a boat or ship into a dock or wharf, to secure it to the wharf
moor added to Dictionary....
Enter word : organ
Enter Primary Meaning : a musical instrument such as a pipe organ or an electric organ
Enter Secondary Meaning : a specific part of the body such as the heart or liver
organ added to Dictionary....
Enter word : pinion
Enter Primary Meaning : A bird's wing, a joint of a bird's wing, a large feather
Enter Secondary Meaning : A small gear with teeth in it, as part of a machine
pinion added to Dictionary....

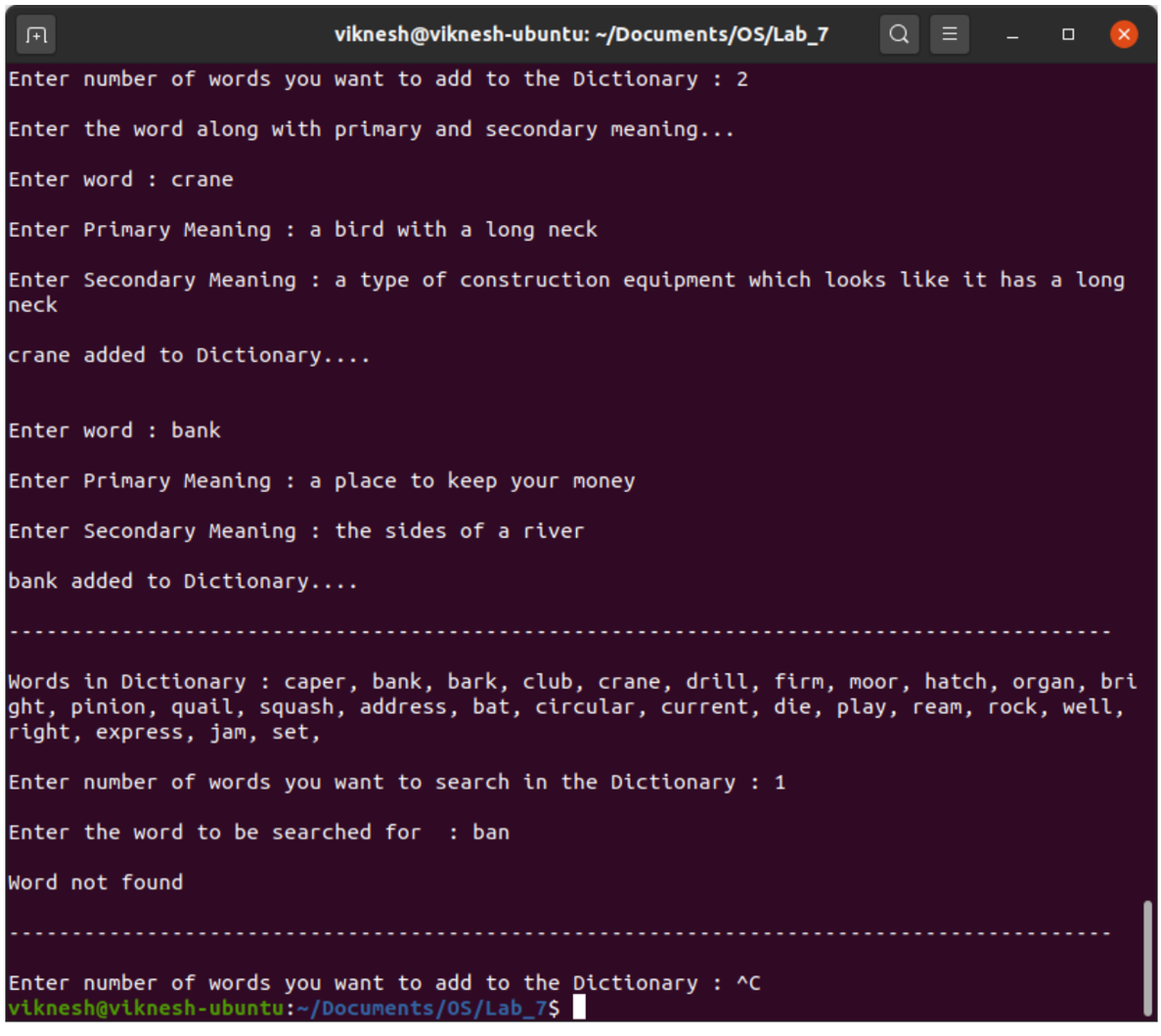

viknesh@viknesh-ubuntu: ~/Documents/OS/Lab_7
Enter word : quail
Enter Primary Meaning : A type of game bird, sometimes prepared for eating
Enter Secondary Meaning : to lose courage, to draw back because of fear
quail added to Dictionary....
Enter word : squash
Enter Primary Meaning : A ball game played by two players with racquets
Enter Secondary Meaning : Compress, crush
squash added to Dictionary....
Enter word : address
Enter Primary Meaning : to speak to
Enter Secondary Meaning : location
address added to Dictionary....
Enter word : bark
Enter Primary Meaning : a tree's out layer
Enter Secondary Meaning : the sound a dog makes
bark added to Dictionary....


viknesh@viknesh-ubuntu: ~/Documents/OS/Lab_7
Enter word : bat
Enter Primary Meaning : an implement used to hit a ball
Enter Secondary Meaning : a nocturnal flying mammal
bat added to Dictionary....
Enter word : circular
Enter Primary Meaning : taking the form of a circle
Enter Secondary Meaning : a store advertisement
circular added to Dictionary....
Enter word : current
Enter Primary Meaning : up to date
Enter Secondary Meaning : flow of water
current added to Dictionary....
Enter word : die
Enter Primary Meaning : to cease living
Enter Secondary Meaning : a cube marked with numbers one through six
die added to Dictionary....


viknesh@viknesh-ubuntu: ~/Documents/OS/Lab_7
Enter word : ream
Enter Primary Meaning : a pile of paper
Enter Secondary Meaning : to juice a citrus fruit
ream added to Dictionary....
Enter word : well
Enter Primary Meaning : in good health
Enter Secondary Meaning : a source for water in the ground
well added to Dictionary....
Enter word : rock
Enter Primary Meaning : a stone
Enter Secondary Meaning : a genre of music
rock added to Dictionary....
Enter word : right
Enter Primary Meaning : correct
Enter Secondary Meaning : direction opposite of left
right added to Dictionary....


viknesh@viknesh-ubuntu: ~/Documents/OS/Lab_7
Enter word : bright
Enter Primary Meaning : very smart or intelligent
Enter Secondary Meaning : filled with light
bright added to Dictionary....
Enter word : express
Enter Primary Meaning : something done fast
Enter Secondary Meaning : to show your thoughts by using words
express added to Dictionary....
Enter word : jam
Enter Primary Meaning : a sweet paste made out of fruit
Enter Secondary Meaning : to put something into a space that is too small for it
jam added to Dictionary....
Enter word : hatch
Enter Primary Meaning : opening in the floor, ceiling, or wall of a ship or aircraft
Enter Secondary Meaning : the process of a baby bird coming out of its egg
hatch added to Dictionary....


viknesh@viknesh-ubuntu: ~/Documents/OS/Lab_7
Enter word : set
Enter Primary Meaning : put something in a position
Enter Secondary Meaning : place where a film or play is made
set added to Dictionary....
Enter word : play
Enter Primary Meaning : To have a good time or do a specific activity
Enter Secondary Meaning : A performance by actors usually shown in theatres
play added to Dictionary....
Words in Dictionary : caper, bark, club, drill, firm, moor, hatch, organ, bright, pinion, quail, squash, address, bat, circular, current, die, play, ream, rock, well, right, expres s, jam, set,
Enter number of words you want to search in the Dictionary : 2
Enter the word to be searched for : cap
Word not found
Enter the word to be searched for : play
Primary Meaning : To have a good time or do a specific activity
Secondary Meaning : A performance by actors usually shown in theatres


viknesh@viknesh-ubuntu: ~/Documents/OS/Lab_7
Enter number of words you want to add to the Dictionary : 2
Enter the word along with primary and secondary meaning...
Enter word : crane
Enter Primary Meaning : a bird with a long neck
Enter Secondary Meaning : a type of construction equipment which looks like it has a long neck
crane added to Dictionary....
Enter word : bank
Enter Primary Meaning : a place to keep your money
Enter Secondary Meaning : the sides of a river
bank added to Dictionary....
Words in Dictionary : caper, bank, bark, club, crane, drill, firm, moor, hatch, organ, bri ght, pinion, quail, squash, address, bat, circular, current, die, play, ream, rock, well, right, express, jam, set,
Enter number of words you want to search in the Dictionary : 1
Enter the word to be searched for : ban
Word not found
Enter number of words you want to add to the Dictionary : ^C
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$

Non-Mandatory (Extra Credits):

5) Trace and understand the working of synchronization algorithms like Dijkstra, Dekker's algorithm.

DIJKSTRA'S BANKER'S ALGORITHM

LOGIC:

- We get the **number of processes** and **number of resources** from the user.
- Next, we get the **MAX Claim table** and **ALLOCATION Resource table** from the user.
- We also get the **Claim vector** from the user.
- Initially, we allocate the resources as per the entries in the Allocation Resource table and calculate the remaining available resources.
- For each process, we **check if that particular process is running** and **whether the number of resources required by the process for its completion is less or equal to the available resources**.
- If **YES**, then we allocate the resources and allow the process to complete and we retrieve back the resources that had been allocated to that process.
- We repeat the previous two steps.
- If all the processes are completed, then the system is in a SAFE state. Otherwise, it results in a deadlock or UNSAFE state.
- Since we **lend the resources** and **get back the resources** once the process completes, this is called **Banker's algorithm**.

C CODE:

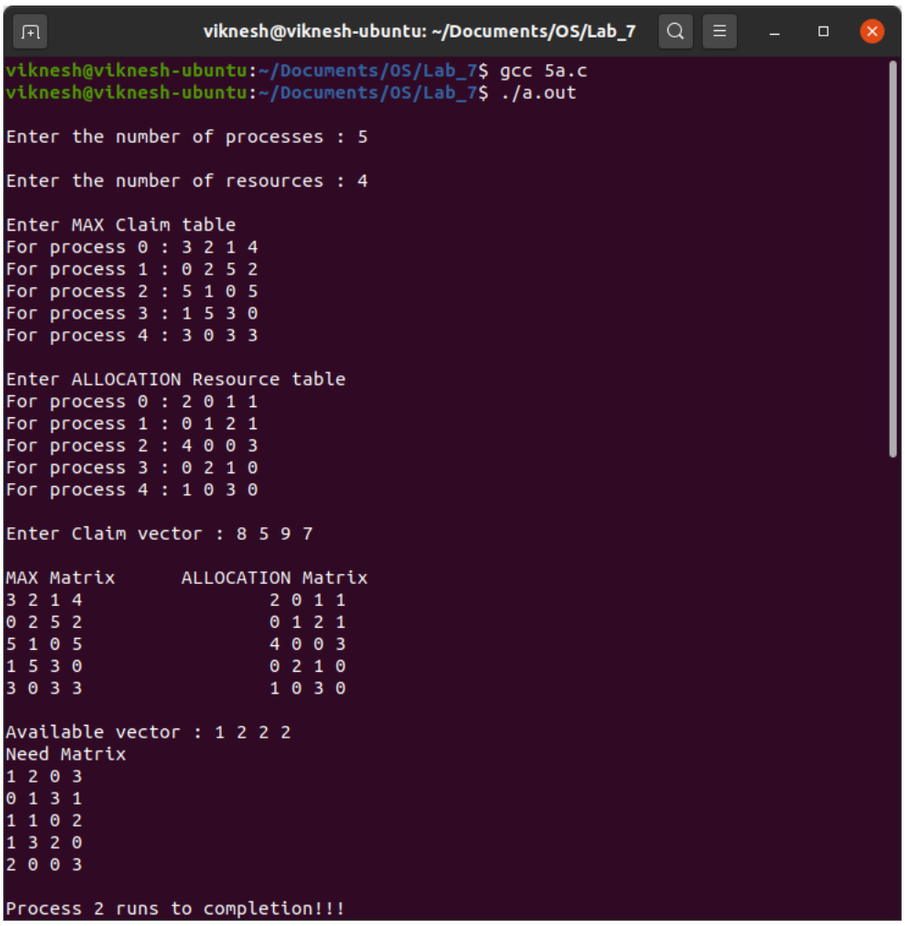
```
#include<stdio.h>
#include<stdbool.h>

#define SIZE 1000

int main()
{
    int p,r;
    int count=0,system=0;
    printf("\nEnter the number of processes : ");
    scanf("%d",&p);
    printf("\nEnter the number of resources : ");
    scanf("%d",&r);

    int max_claim[p][SIZE];
    int curr[p][SIZE];
    int avl[r];
    int alloc[p];
    int max_res[r];
    int running[p];
    int safe_sequence[p];
    printf("\nEnter MAX Claim table\n");
    for(int i=0;i<p;++i)
    {
            printf("For process %d : ",i);
            for(int j=0;j<r;++j)
```

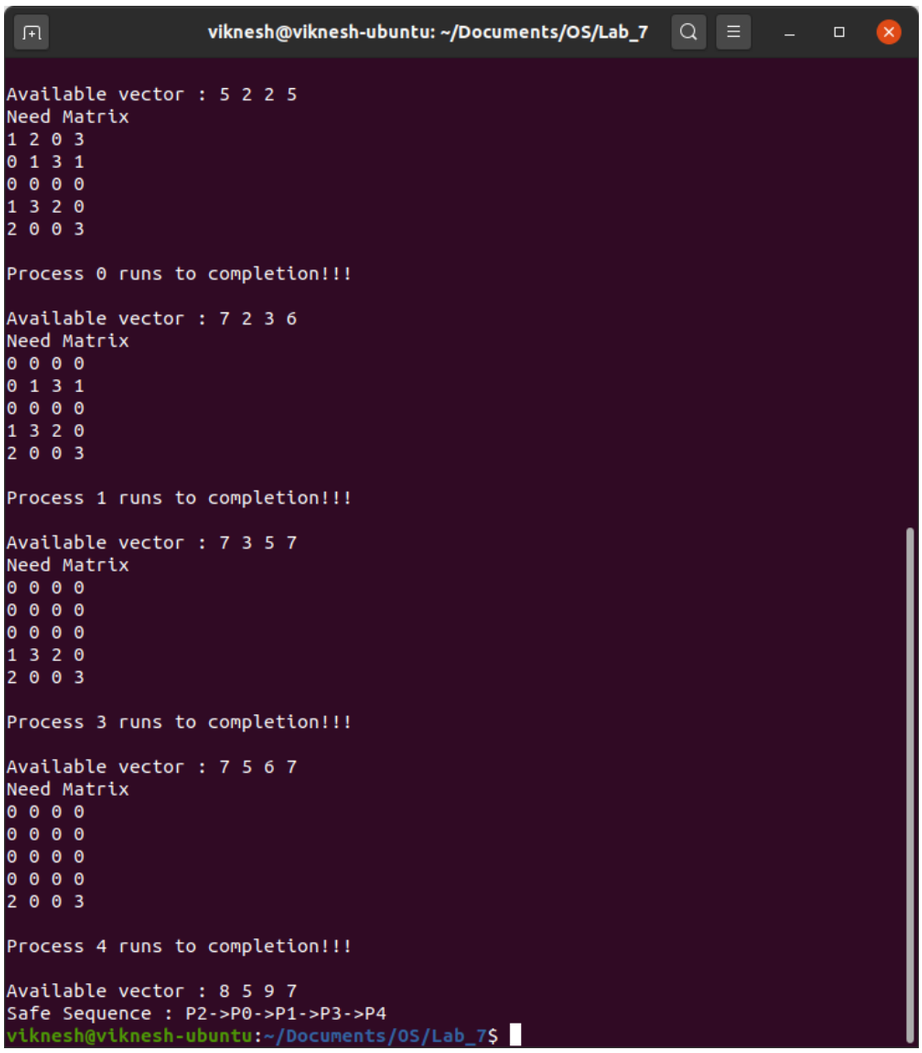
```
		{
			scanf("%d",&max_claim[i][j]);
		}
		running[i]=1;
		count++;
}

printf("\nEnter ALLOCATION Resource table\n");
for(int i=0;i<p;++i)
{
		printf("For process %d : ",i);
		for(int j=0;j<r;++j)
		{
			scanf("%d",&curr[i][j]);
		}
}

printf("\nEnter Claim vector : ");
for(int i=0;i<r;++i)
{
		scanf("%d",&max_res[i]);
}

printf("\nMAX Matrix\tALLOCATION Matrix\n");
for(int i=0;i<p;++i)
{
		for(int j=0;j<r;++j)
		{
			printf("%d ",max_claim[i][j]);
		}
		printf("\t\t");
		for(int j=0;j<r;++j)
		{
			printf("%d ",curr[i][j]);
		}
		printf("\n");
}

for(int i=0;i<p;++i)
{
		for(int j=0;j<r;++j)
		{
			alloc[j]+=curr[i][j];
		}
}

for(int i=0;i<r;++i)
{
		avl[i]=max_res[i]-alloc[i];
}

while(count!=0)
{
		bool safe=false;
		printf("\nAvailable vector : ");
		for(int i=0;i<r;++i)
		{
			printf("%d ",avl[i]);
		}
```

```
        printf("\nNeed Matrix\n");
        for(int i=0;i<p;++i)
        {
                if(running[i]==1)
                {
                        for(int j=0;j<r;++j)
                        {
                                printf("%d ",max_claim[i][j]-curr[i][j]);
                        }
                }
                else
                {
                        for(int j=0;j<r;++j)
                        {
                                printf("0 ");
                        }
                }
                printf("\n");
        }

        for(int i=0;i<p;++i)
        {
                if(running[i]==1)
                {
                        int flag=1;
                        for(int j=0;j<r;++j)
                        {
                                if(max_claim[i][j]-curr[i][j]>avl[j])
                                {
                                        flag=0;
                                        break;
                                }
                        }

                        if(flag==1)
                        {
                                printf("\nProcess %d runs to completion!!!\n",i);
                                running[i]=0;
                                safe_sequence[p-count]=i;
                                --count;
                                safe=true;

                                for(int j=0;j<r;++j)
                                {
                                        avl[j]+=curr[i][j];
                                }
                                break;
                        }
                }
        }

        if(!safe)
        {
                printf("\nThe processes are in UNSAFE state!!!");
                system=1;
                break;
        }
}
```

```
    printf("\nAvailable vector : ");
    for(int i=0;i<r;++i)
    {
            printf("%d ",avl[i]);
    }

    if(system==0)
    {
            printf("\nSafe Sequence : ");
            for(int i=0;i<p-1;++i)
            {
                    printf("P%d->",safe_sequence[i]);
            }
            printf("P%d\n",safe_sequence[p-1]);
    }

    return 0;
}
```

OUTPUT:





DEKKER'S ALGORITHM

LOGIC:

- We only run two threads.
- We ask the user to enter the number of times each thread must run.
- We use favoured thread semaphore to enter the critical section.
- This avoids deadlock problem.

C CODE:

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

int favoured_thread=0;
bool thread_enter[2];
int COUNT=1;

void *thread_0()
{
    for(int i=0;i<COUNT;++i)
    {
            thread_enter[0]=true;
            while(thread_enter[1]==true)
            {
                    if(favoured_thread==1)
```

```
                {
                        thread_enter[0]=false;
                }
                while(favoured_thread==1)
                {
                        ;
                }
                thread_enter[0]=true;
        }

        printf("\nThread 0 in Critical Section");

        favoured_thread=1;
        thread_enter[0]=false;
    }
}

void *thread_1()
{
    for(int i=0;i<COUNT;++i)
    {
        thread_enter[1]=true;
        while(thread_enter[0]==true)
        {
                if(favoured_thread==0)
                {
                        thread_enter[1]=false;
                }
                while(favoured_thread==0)
                {
                        ;
                }
                thread_enter[1]=true;
        }

        printf("\nThread 1 in Critical Section");

        favoured_thread=0;
        thread_enter[1]=false;
    }
}

int main()
{
    pthread_t tid[2];
    thread_enter[0]=false;
    thread_enter[1]=false;
    printf("\nEnter the number of times to run the threads : ");
    scanf("%d",&COUNT);
    pthread_create(&tid[0],NULL,thread_0,NULL);
    pthread_create(&tid[1],NULL,thread_1,NULL);
    pthread_join(tid[0],NULL);
    pthread_join(tid[1],NULL);
    printf("\n");
    return 0;
}
```

OUTPUT:

```
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$ gcc 5b.c -pthread
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$ ./a.out

Enter the number of times to run the threads : 5

Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$
```

```
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$ gcc 5b.c -pthread
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$ ./a.out

Enter the number of times to run the threads : 15

Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
Thread 0 in Critical Section
Thread 1 in Critical Section
Thread 1 in Critical Section
Thread 0 in Critical Section
viknesh@viknesh-ubuntu:~/Documents/OS/Lab_7$
```