**LAB-5**

**1.PRODUCER-CONSUMER**

#include<stdio.h>

#include<conio.h>

int mutex=1;

int full=0;

int empty=10;

int cnt=0;

int wait(int s)

{

while(s<=0);

s--;

return s;

}

int signal(int s)

{

s++;

return s;

}

void producer()

{

empty=wait(empty);

mutex=wait(mutex);

cnt++;

printf("Producer produces an item %d\n",cnt);

mutex=signal(mutex);

full=signal(full);

}

void consumer()

{

full=wait(full);

mutex=wait(mutex);

printf("Consumer consumes an item %d\n",cnt);

cnt--;

mutex=signal(mutex);

empty=signal(empty);

}

void main()

{

int choice;

printf("1.Produce\n2.Consume\n3.Exit\n");

while(1)

{

printf("Enter your choice:\n");

scanf("%d",&choice);

switch(choice)

{

case 1:if(empty==0)

{

printf("Buffer is full\n");

}

else{

producer();

}

break;

case 2:if(full==0)

{

printf("Buffer is empty\n");

}

else{

consumer();

}

break;

case 3:exit(0);

break;

default:printf("Invalid choice\n");

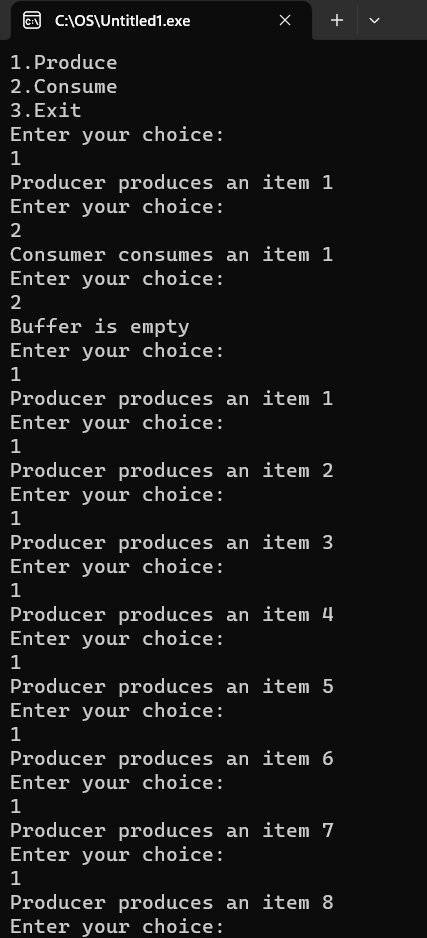
}

}

getch();

}

OUTPUT



**2.DINING PHILOSOPHER**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#define N 5

#define THINKING 2

#define HUNGRY 1

#define EATING 0

#define LEFT (phnum + 4) % N

#define RIGHT (phnum + 1) % N

int state[N];

int phil[N] = { 0, 1, 2, 3, 4 };

sem\_t mutex;

sem\_t S[N];

void test(int phnum)

{

if (state[phnum] == HUNGRY

&& state[LEFT] != EATING

&& state[RIGHT] != EATING) {

// state that eating

state[phnum] = EATING;

sleep(2);

printf("Philosopher %d takes fork %d and %d\n",

phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is Eating\n", phnum + 1);

// sem\_post(&S[phnum]) has no effect

// during takefork

// used to wake up hungry philosophers

// during putfork

sem\_post(&S[phnum]);

}

}

// take up chopsticks

void take\_fork(int phnum)

{

sem\_wait(&mutex);

// state that hungry

state[phnum] = HUNGRY;

printf("Philosopher %d is Hungry\n", phnum + 1);

// eat if neighbours are not eating

test(phnum);

sem\_post(&mutex);

// if unable to eat wait to be signalled

sem\_wait(&S[phnum]);

sleep(1);

}

// put down chopsticks

void put\_fork(int phnum)

{

sem\_wait(&mutex);

// state that thinking

state[phnum] = THINKING;

printf("Philosopher %d putting fork %d and %d down\n",

phnum + 1, LEFT + 1, phnum + 1);

printf("Philosopher %d is thinking\n", phnum + 1);

test(LEFT);

test(RIGHT);

sem\_post(&mutex);

}

void\* philosopher(void\* num)

{

while (1) {

int\* i = num;

sleep(1);

take\_fork(\*i);

sleep(0);

put\_fork(\*i);

}

}

int main()

{

int i;

pthread\_t thread\_id[N];

// initialize the semaphores

sem\_init(&mutex, 0, 1);

for (i = 0; i < N; i++)

sem\_init(&S[i], 0, 0);

for (i = 0; i < N; i++) {

// create philosopher processes

pthread\_create(&thread\_id[i], NULL,

philosopher, &phil[i]);

printf("Philosopher %d is thinking\n", i + 1);

}

for (i = 0; i < N; i++)

pthread\_join(thread\_id[i], NULL);

}

OUTPUT

