#include<stdio.h>

#include<stdlib.h>

int mutex=1,full=0,empty,item=0;

int wait(int x);

int signal(int x);

void producer();

void consumer();

int main()

{

int n;

printf("\nEnter Total Number of Items:");

scanf("%d",&n);

empty = n;

while(1)

{

printf("\n1.Producer\n2.Consumer\n3.Exit\n");

printf("\nEnter your choice:");

scanf("%d",&n);

switch(n)

{

case 1: if((mutex==1)&&(empty!=0))

producer();

else

printf("Buffer is full!!");

break;

case 2: if((mutex==1)&&(full!=0))

consumer();

else

printf("Buffer is empty!!");

break;

case 3:

printf("\nExit...........");

exit(0);

break;

}

}

return 0;

}

int wait(int x)

{

return (--x);

}

int signal(int x)

{

return(++x);

}

void producer()

{

mutex=wait(mutex);

full=signal(full);

empty=wait(empty);

item++;

printf("\nProducer produces the item %d",item);

mutex=signal(mutex);

}

void consumer()

{

mutex=wait(mutex);

full=wait(full);

empty=signal(empty);

printf("\nConsumer consumes item %d",item);

item--;

mutex=signal(mutex);

}

ALGORITHM SEMAPHORE(Max\_Items)

STEP 1: Declare variables mutex := 1, full:=0, empty:=Max\_Items, item=0;

STEP 2: If Producer is called it checks if mutex is 1 and empty is not equal to zero then

                   it decreases the value of mutex to 0 and locks the Critical section so that only Producer can produce item by incrementing full and decrementing empty by one. After Item is produced mutex again get                                   incremented to one.

            Else

                   The buffer is Full.

STEP 3: If Consumer is called it checks if mutex is 1 and full is not equal to zero then

                    it decreases the value of mutex to 0 and locks Critical Section so that only consumer can consume item by decrementing full and incrementing empty by one. After Item is Consumed mutex again                                        get incremented to one.

             Else

                    The buffer is Empty.

STEP 4: Exit.