PRODUCER CONSUMER PROBLEM.

AIM:

PROGRAM TO IMPLEMENT THE PRODUCER AND CONSUMER PROBLEM USING C LANGUAGE.

Producer consumer problem is also known as bounded buffer problem. In this problem we have two processes, producer and consumer, who share a fixed size buffer.

Producer work is to produce data or items and put in buffer. Consumer work is to remove data from buffer and consume it.

We have to make sure that producer do not produce data when buffer is full and consumer do not remove data when buffer is empty.

ALGORITHM:

Step 1: Start

Step 2: Define the maximum buffer size.

Step 3:Enter the number of producers and consumers.

Step 4: The producer produces the job and put it in the buffer.

Step 5:The consumer takes the job from the buffer.

Step 6:If the buffer is full the producer goes to sleep.

Step 7:If the buffer is empty then consumer goes to sleep.

Step 8:Stop

Write a C Program to implement Producer Consumer Problem.

For example:

Test	Input	Result
T1	1 2 3	1.Producer 2.Consumer 3.Exit Producer produces the item 1 Consumer consumes item 1

PROGRAM:

```
#include<stdlib.h>
int mutex=1, full=0,empty=3,x=0;
int main()
{
    int n;
    void producer();
    void consumer();
    int wait(int);
    int signal(int);

printf("1.Producer\n2.Consumer\n3.Exit");
    for(int i=1;i>0;i++){
        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;
```

```
27          case 3:
28                  exit(0);
29                  break;
30          }
31      }
32      return 0;
```

```
33  }
34  int wait(int s)
35  {
36      return(--s);
37  }
38  int signal(int s)
39  {
40      return(++s);
41  }
```

```
42  void producer()
43  {
44      mutex = wait(mutex);
45      full = signal(full);
46      empty = wait(empty);
47      x++;
48      printf("\nProducer produces the item %d",x);
49      mutex=signal
50
51      (mutex);
```

```
52  }
53  void consumer()
54  {
55      mutex = wait(mutex);
56      full = wait(full);
57      empty = signal(empty);
58      printf("\nConsumer consumes item %d",x);
59      x--;
60      mutex = signal(mutex);
61  }
```

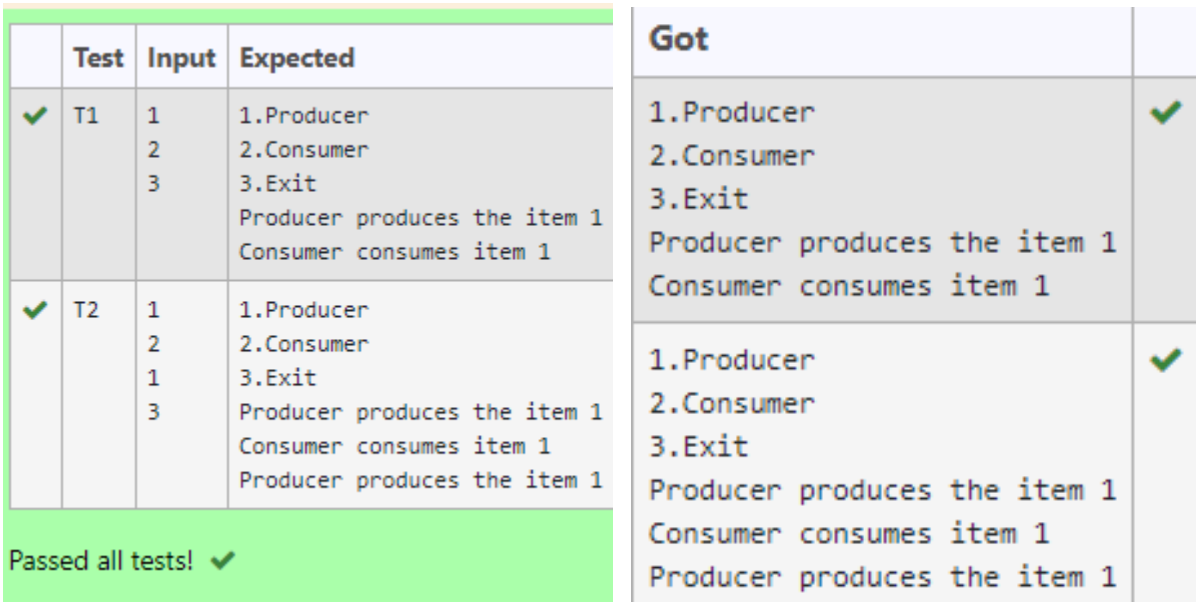
CODE:

```
#include<stdio.h>
#include<stdlib.h>
int mutex=1;
int full=0;
int empty=10,x=0;
void producer()
{
--mutex;
++full;
--empty;
x++;
++mutex;
printf("\nProducer produces the item 1");
}
void consumer()
{
--mutex;
--full;
++empty;
x--;
++mutex;
printf("\nConsumer consumes item 1");
```

```
}
int main(){
int n;
printf("1.Producer"
"\n2.Consumer"
"\n3.Exit"
);
for(int i=1;i>0;i++){
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:
exit(0);
```

```
break;
}
}
}
```

RESULT:



	Test	Input	Expected	Got	
✔	T1	1 2 3	1.Producer 2.Consumer 3.Exit Producer produces the item 1 Consumer consumes item 1	1.Producer 2.Consumer 3.Exit Producer produces the item 1 Consumer consumes item 1	✔
✔	T2	1 2 1 3	1.Producer 2.Consumer 3.Exit Producer produces the item 1 Consumer consumes item 1 Producer produces the item 1	1.Producer 2.Consumer 3.Exit Producer produces the item 1 Consumer consumes item 1 Producer produces the item 1	✔

Passed all tests! ✔

RESULT:

PRODUCER CONSUMER PROGRAM WAS SUCCESSFULLY IMPLEMENT USING C LANGUAGE.