Lab 5 : Write a C program to stimulate Producer-Consumer problem using semaphores

19-1-23

```
#include <stdio.h>
#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("\n1. Producer \n2. Consumer \n3. Exit");
  while(1)
  {
    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: exit(0);
              break;
    }
  }
```

```
    return 0;
}
int wait (int s)
{
    return (--s);
}
Void producer ()
{
    mutex = wait (mutex);
    full = signal (full);
    empty = wait (empty);
    x++;
    printf ("\n Producer produces the item %d", x);
    mutex = signal (mutex);
}
Void consumer ()
{
    mutex = wait (mutex);
    full = wait (full);
    empty = signal (empty);
    printf ("\n Consumer consumes item %d", x);
    x--;
    mutex = signal (mutex);
}
```

Output:

1) Producer
2) Consumer
3) Exit

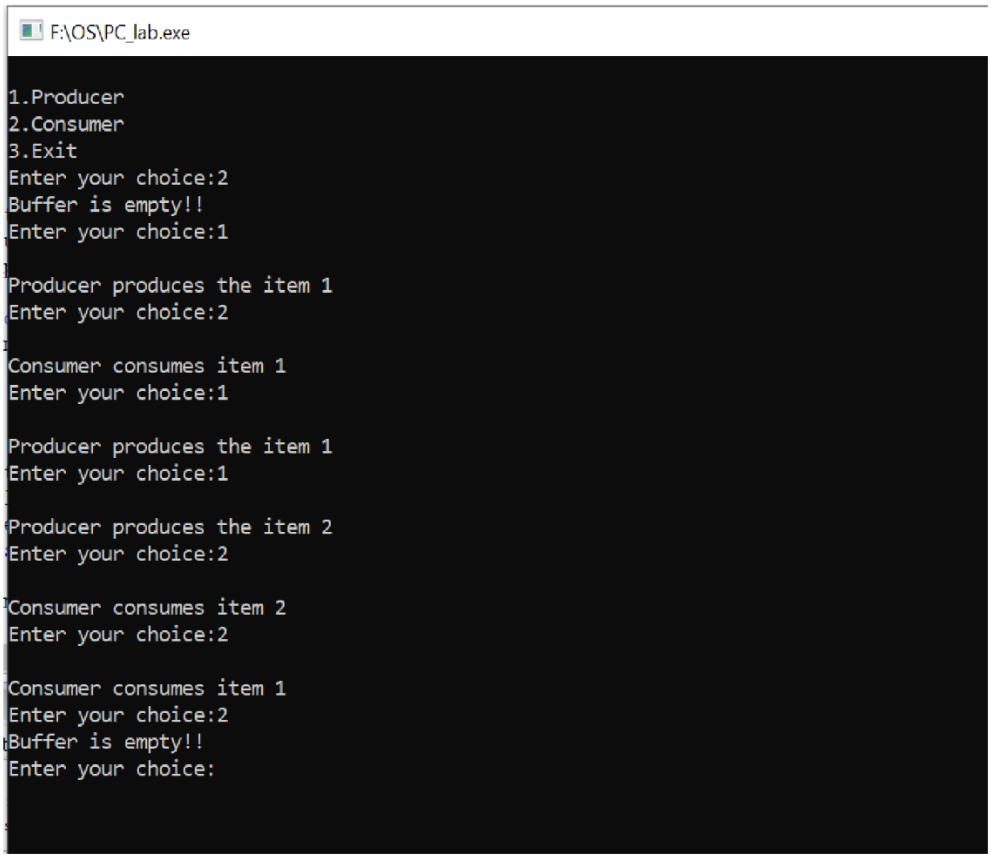
Enter your choice : 1

Producer produces the item 1

Enter your choice : 1
Producer produces the item 2
Enter your choice : 1
Producer produces the item 3
Enter your choice : 1
Buffer is full !!
Enter your choice : 2
Consumer consumer item 3
Enter your choice : 2
Consumer consumes item 2
Enter your choice : 2
Consumer consumes item 1
Enter your choice : 2
Buffer is empty !!

26/7/23
10/10

Output:


F:\OS\PC_lab.exe
1.Producer
2.Consumer
3.Exit
Enter your choice:2
Buffer is empty!!
Enter your choice:1
Producer produces the item 1
Enter your choice:2
Consumer consumes item 1
Enter your choice:1
Producer produces the item 1
Enter your choice:1
Producer produces the item 2
Enter your choice:2
Consumer consumes item 2
Enter your choice:2
Consumer consumes item 1
Enter your choice:2
Buffer is empty!!
Enter your choice:

Earliest deadline first scheduling

```
#include <stdio.h>
#include <stdlib.h>
int et[10], i, n, dl[10], p[10], ready[10], flag=1;
int lcm(int a, int b){
    int max = (a>b) ? a : b;
    while (1){
        if (max % a == 0 && max % b == 0)
            return max
int hyperperiod(float period[], int n){
    int x = period[0];
    n--;
```

```
while (n >= 1){
    k = lcm (k, period [n--]); }
    return k; }

int edf (float *period, int n, int t, float *deadline){
    int i, small = 10000.0f, smallindex = 0;
    for (i = 0; i < n; i++){
    if (period[i] < small && (period[i]-t) <= deadline[i]){
    int small = period[i];
        smallindex = i; }}
    if (small == 10000.0f)
    return -1;
    return smallindex; }

int main()
{
    int i, n, c, d, k, j, ncalltime = 0, time = 0, task,
    preemption_count;
    float exec[20], period[20], individual_util[20], flag[20],
    release[20], deadline[20], instance[20], ex[20],
    responsemax[20], responsemin[20], tempmax;
    float util = 0;
    printf("\n Earliest Deadline first Algorithm\n");
    FILE *read;
    read = fopen("Sampledata.doc", "r");
    fscanf(read, "%d", &n)
    for (i = 0; i < n; i++)
    {
```

```
    fscanf(read, "%f", &release[i]);
    fscanf(read, "%f", &period[i]);
    fscanf(read, "%f", &deadline[i]);
}
fclose(read);
for(i=0; i<n; i++)
{
    individual_util[i] = exec[i]/period[i];
    deadline[i] = period[i];
}
util = util*100;
if(util>100)
    printf("\n Utilisation factor = %0.2f \n");
else
{
    c=0;
    while(time<k)
    {
    real time = time+1;
    if(task==-1)
    {
    printf("-");
    time++;
    continue;
}
```

```
instance[task]++;
printf("T%d", task);
exc[c++] = task;
if(instance[task] == exec[task])
{ temp max = next time - (period[task] - deadline[task]);
else{
response min[task] = instance[task];
}
for(i=0; i<n; i++)
{ printf("\n Maximum Response time of Task %d = %f",
    i, response max[i]);
  printf("\n Minimum Response time of Task %d = %f",
    i, response min[i]);
}
preemption_count = 0;
for(i=0; i<k; i++)
{ flag[i] = 1;
  d = exc[i];
  for(j=i+1; d == exc[j]; j++)
  flag[d]++;
  if(flag[d] == cexec[d])
  flag[d] = 1;
  else
  { flag[d]++;
    preemption_count++;
  }
}
printf("\n Preemption Count = %d", preemption_count);
}
return 0;
}
```

Output

Enter the number of processes : 3

Enter execution times : 1 5 7

Enter deadlines : 2 4 7

~~0 P1 1~~

0 P1 1 PR 2 P1 3 PR 4 P1 5 PR 6 P1 7 PR 8
P1 9 PR 10 P1 11 P3 12 P1 13 P2 14 P1 15 PR
16 P1 17 PR 18 P1 19 PR 20 P1 21 PR 22 P1
23 P3 24 P1 25 PR 26 P1 27 PR 28 P1

10/10

M
25/7/23

ab 6

) Write

#includ

#includ

Void

{

int a

int

int

int

int

char

print

Scan

for

fini

Prin

for

fo

Output:

Output: