**Lab Exercise 6**

**Producer Consumer Using OpenMP**

**Name:** Ayush Sharma

**Reg. No:** 15BCE1335

**Faculty:** Prof. Gayatri. R

Use OpenMP to implement the Producer-Consumer problem in which some of the threads are producers and other are the consumers. The producer insert text in the shared queue. Before inserting any text producer will check the shared queue whether the similar text present in the memory or not. The consumers are not allowed to access the shared queue while producer are completing the task. Consumer will read the text from the shared queue will be displayed and will be removed from the shared queue.

**Theory:**

My program creates four threads, two of which are producer and other two are consumers. Each producer place a character in the shared queue check for duplication and consumer will get the text from the shared queue, print it, and remove it too. This will help in synchronization of producer and consumer.

**Code:**

char buffer[SIZE];

int nextin = 0;

int nextout = 0;

int count = 0;

int empty = 1;

int full = 0;

int i,counter;

void put(char item)

{

buffer[nextin] = item;

nextin = (nextin + 1) % SIZE;

count++;

if (count == SIZE)

full = 1;

if (count == 1)

empty = 0;

}

void producer(int tid)

{

char item;

while( i < NUMITER)

{

#pragma omp critical

{

item = 'A' + (i % 26);

put(item);

i++;

printf("%d Producing %c ...\n",tid, item);

}

sleep(1);

}

}

char get()

{

char item;

item = buffer[nextout];

nextout = (nextout + 1) % SIZE;

count--;

if (count == 0) // buffer is empty

empty = 1;

if (count == (SIZE-1))

// buffer was full

full = 0;

return item;

}

void consumer(int tid)

{

char item;

while(j < NUMITER )

{

#pragma omp critical

{

j++;

item = get();

printf("%d ...Consuming %c\n",tid, item);

}

sleep(1);

}

}

int main()

{

int tid;

i=j=0;

#pragma omp parallel firstprivate(i,j) private(tid) num\_threads(4)

{

tid=omp\_get\_thread\_num();

if(tid%2==1)

producer(tid);

else

consumer(tid);

}

}

**Screenshot:**

