Task – 1

The output of the code was diffreent every time it was run. The output ranged from 8 to 15 with frequent values as 10,11,12.

Below is the modified code. It produced output 15. I modified the code in the same way as the example was provided. Testset function solved the problem by limiting the access to critical section.

const int m=5;

int n;

int lock=0;

atomic int testset(int &i)

{

if (i == 0) {

i = 1;

return 1;

}

else

return 0;

}

void incr(char id){

int j=1;

while(j <= m)

{

while(testset(lock))

{

n = n + 1;

j++;

lock=0;

break;

}

}

}

main()

{

n = 0;

cobegin

{

incr('A'); incr('B'); incr('C');

}

cout << "The sum is " << n << endl;

}

Task – 2

I ran the given code, and it was giving value of n ranging from 10-18 and most of the times, the output was 15,16. This was happening as two different functions running on two different processes was accessing the variable n. to avoid this situation I used testset function with lock variable as controling variable to control access to the variable n.

The modified code is as below, and it produced the output of 20 everytime I exceuted it.

int n = 0;

int lock=0;

atomic int testset(int &i)

{

if (i == 0) {

i = 1;

return 1;

}

else

return 0;

}

void p()

{

int temp, i=0;

while (i < 10) {

while(testset(lock))

{

temp = n;

n = temp + 1;

i++;

lock=0;

break;

}

}

}

void q() {

int temp, i=0;

while (i < 10) {

while(testset(lock))

{

temp = n;

n = temp + 1;

i++;

lock=0;

break;

}

}

}

void main() {

cobegin { p(); q(); }

cout << "The value of n is " << n << "\n";

}

Task – 3

This code seems to be implementing mutual exclusion by changing the value of Bolt variable but it can not be done in this way. Just chaning/controling the value of one variable and using if-else can not result in mutual exclusion.

The output ranges from 24-28 most of the time.

Task – 4

This program is doing mutual exclusion, where each process before entering into critcal section, checks that either some other process is in the critical section or not. If there is no other process in the critical section, then Process A enters the critical section and sets a variable which helps other processes to know that the process A is in critical section. If some other process let say B isin the critical section, then A keeps on waiting until B gets out of the cricitcal section