**CSCI465 Operating System**

**Assignment 4**

**Course outcomes: 4**

**Question 1:**

1. What is the disadvantages of **Peterson’s solution**?
2. What do we mean by **bounded waiting** and does Semaphore provide **bounded waiting**?
3. What happens if a critical section algorithm does not implement **progress**?
4. Describe a scenario in which we may have a **Race Condition**.

Solution:

1\_ it involves busy waiting, And It is limited to 2 processes.

2\_ there exists a bound, or limit, on the number of times other processes are allowed to enter their critical sections after a process has made request to enter its critical section and before that request is granted.

3\_ that is mean if we have 2 process the first proses can execute forever without letting the second proses enter its critical section at all.

4\_ A race condition occurs when two or more threads can access shared data and they try to change it at the same time.

**Question 2:**

Write a java program that will create two processes T1 and T2. T1 will ask the user to enter an integer X and then T1 will add 2 to X and T2 will subtract 4 from X. Afterwards, T2 will print the result of X. Use Semaphores to synchronize the operations of the two processes.

package thread;  
  
  
public class threads extends Thread {  
    int x=0,y;  
  
    public threads(int x) {  
        this.x=x;  
    }  
    public void run(){  
    x+=2;  
    y=x-4;  
    }  
  
    public int getY() {  
        return y;  
    }      
}

package thread;  
  
import java.util.Scanner;  
public class Thread {

    public static void main(String[] args) {  
Scanner s=new Scanner(System.in)        ;  
        System.out.println("enter x :");  
int x=s.nextInt();  
threads t1= new threads(x);  
threads t2= new threads(x);  
  
t1.run();  
t2.run();  
        System.out.println(t2.getY());  
    }   
}