1. Let \( x \) and \( y \) be two shared variables stored in the main memory. Assuming that the two variables are initialized to 10, what can be the possible outcomes for \( x \) and \( y \) if the following two processes are executed concurrently?

\[
\text{Process A: } x = x + y \quad \text{Process B: } y = x \times x
\]

2. Draw a precedence graph (a.k.a., dependency graph) that reflects the parallelism in the following code. Recall a precedence graph is a directed graph in which the nodes (vertices) represent processes and the edges represent the causal relation between two processes. If an edge exists from process node \( P_i \) to \( P_j \) then the process \( P_j \) must start executing its code \( c_j \) only when process \( P_i \) finishes executing its code \( c_i \). Also we use the notation \texttt{ParBegin} and \texttt{ParEnd} to indicate a block in which program codes (or instructions) must run in parallel. Another alternative notation that can be found in the literature is \texttt{CoBegin} and \texttt{CoEnd} for concurrent begin/end.

\[
\begin{align*}
\text{Begin} \\
& c_0; \\
\text{ParBegin} \\
& c_1; \\
\text{Begin} \\
& c_2; \\
& c_3; \\
& c_4; \\
\text{End} \\
& c_5; \\
& c_6; \\
\text{Begin} \\
& c_7; \\
& c_8; \\
& c_9; \\
\text{End} \\
\text{ParEnd} \\
& c_{10}; \\
\text{End}
\end{align*}
\]
3. Complete the following program code to reflect the process precedence graph shown below (where \( c_i \) refers to the code-i). This code basically consists of creating two processes in parallel: the first process executes program \( P_1 \) and the second process executes program \( P_2 \). You should use one semaphore for that.

```
Begin
  ParBegin \( P_1; \ P_2; \) ParEnd
End
```

Program \( P_1 \):
```
Begin
  code_1;
End
```

Program \( P_2 \):
```
Begin
  code_2;
End
```

4. Complete the following program code to reflect the process precedence graph shown below (where \( c_i \) refers to the code-i). This code basically consists of creating two processes in parallel: the first process executes program \( P_1 \), and the second process executes program \( P_2 \). You should use semaphores for that.

```
Begin
  ParBegin \( P_1 \); \( P_2 \); ParEnd
End
```

Program \( P_1 \):
```
Begin
  code_1;
End
```

Program \( P_2 \):
```
Begin
  code_2;
End
```

```
5. Let Process $P_1$ and $P_2$ be two processes that execute concurrently using a shared semaphore $s$ initialized to 10. Each process increments, in an infinite loop, an integer $i$ (process $P_1$) or $j$ (process $P_2$) both initialized to 0. After executing for a while, the values of $i$ and $j$ will change considerably. In this case, what will be the relation between both variables?

Begin

Semaphore $s$;
Integer $i$, $j$;
$s=10$; $i=0$; $j=0$;
ParBegin $P_1$; $P_2$; ParEnd
End

Program $P_1$: Program $P_2$: Begin Begin
  $code_1$; $code_2$;
  $code_3$; End

Begin End
while(1) while(1);
{
  Acquire($s$); j++; 
  i++; Release($s$);
}

End

a) $i = j$      b) $j = i+10$      c) $i \leq j$      c) $i = j + 10$      d) $i \geq j + 10$
6. Give the precedence graph for the following code and check whether a deadlock may occur or not. If there is a deadlock, then modify the codes to fix that:

```
Begin
    Semaphore s1, s2;
    s1 = 0; s2 = 0;
    ParBegin P1; P2; ParEnd
End
```

Program P1:
```
Begin
    code1;
    Release(s2);
End
```

Program P2:
```
Begin
    Acquire(s1);
    code2;
    Release(s1);
End
```

7. Modify the following code (using two semaphores) so that the two programs P1 and P2 execute alternatively (i.e., ..., P1, P2, P1, P2, P1, ... with P1 being the first to start execution.

```
Begin
    ParBegin P1; P2; ParEnd
End
```

Program P1:
```
Begin
    while (true) {code1;}
End
```

Program P2:
```
Begin
    while (true) {code2;}
End
```

Copyright © Karim Lounis, 2019.
8. In the previous program, the two programs $P_1$ and $P_2$ are executed in such a way fairness is guaranteed. Modify the previous program (using two semaphores) so that the execution order becomes ..., $P_1, P_2, P_2, P_1, P_2, P_1, ...$ with $P_2$ being the first to start execution.

9. Consider the following notation: let $Bc_1$ (for Begin $c_1$) expresses the starting of the code $c_1$ and $Ec_1$ (for End $c_1$) the ending of the code $c_1$, and let the precedence graph shown below reflects the execution order of program codes $c_1, c_2, c_3,$ and $c_4$. Then, do the execution sequences shown below respect the precedence graph constraints or not?

![Precedence Graph](image_url)

Copyright © Karim Lounis, 2019.
10. Write the program code that reflects the following precedence graph using six processes $P_1, \ldots, P_6$ each executing its dedicated code $c_1, \ldots, c_6$. You are required to use only two semaphores.

\begin{center}
\begin{tikzpicture}
  \node (c1) at (0,0) {$c_1$};
  \node (c2) at (-2,-2) {$c_2$};
  \node (c3) at (0,-2) {$c_3$};
  \node (c4) at (2,-2) {$c_4$};
  \node (c5) at (-2,-4) {$c_5$};
  \node (c6) at (2,-4) {$c_6$};

  \draw[->] (c2) -- (c1);
  \draw[->] (c3) -- (c2);
  \draw[->] (c4) -- (c3);
  \draw[->] (c5) -- (c4);
  \draw[->] (c6) -- (c5);

\end{tikzpicture}
\end{center}

\begin{itemize}
  \item a) $Bc_1 Ec_1 Bc_2 Ec_2 Ec_3 Bc_3 Ec_4 Bc_4$
  \item b) $Bc_1 Bc_2 Ec_2 Ec_1 Bc_3 Bc_4 Ec_3 Ec_4$
  \item c) $Bc_1 Bc_2 Ec_1 Ec_2 Bc_3 Bc_4 Ec_3 Ec_4$
  \item d) $Bc_1 Bc_2 Ec_2 Ec_1 Bc_3 Ec_3 Bc_4 Ec_4$
  \item e) $Bc_1 Bc_2 Ec_2 Ec_3 Ec_1 Bc_4 Ec_4 Ec_3$
  \item f) $Bc_1 Bc_2 Ec_2 Ec_3 Ec_3 Ec_1 Bc_4 Ec_4$
  \item g) $Bc_1 Bc_2 Ec_1 Ec_2 Ec_3 Ec_3 Bc_4 Ec_4$
  \item h) $Bc_2 Ec_2 Ec_1 Bc_4 Ec_4 Ec_3 Bc_3 Ec_3$
\end{itemize}