Note: You may NOT look on the Internet for solutions. You may NOT ask for solutions on the Internet. You may NOT GIVE or SHOW any FILES to anyone. You may NOT RECEIVE or LOOK AT any FILES from anyone. You MAY discuss the problems in ***high level*** terms—you may NOT dictate code or pseudocode or give answers away. You MUST write the solutions ON YOUR OWN. See "Academic Integrity" policy on the syllabus.

P1 Write the following methods for the clique hill climbing algorithm:

(a). `getRandomset(n, k)` (10 points)

- Inputs: Integers n and k
- Returns: A list of k random, distinct elements from {0, 1, 2, ..., n-1} (in sorted order)

(b). `getNumEdges(set, g)` (10 points)

- Inputs:
  - set = a list of distinct vertices
  - g = an undirected graph
- Returns: The number of pairs of adjacent vertices in that list

---

(c). `getBestNeighbor(n, k, set, g)` (25 points)

- Inputs:
  - n = the number of vertices in graph "g"
  - k = the number of vertices in list "set"
  - set = a list of k distinct vertices
  - g = an undirected graph
- Returns: [bestNeighbor, numEdges, vertex_eliminated, vertex_added], where:
  - bestNeighbor = a list of k distinct vertices that differs from "set" in exactly one vertex and that maximizes the number of edges
  - numEdges = the number of pairs of adjacent vertices in "bestNeighbor"
  - vertex_eliminated = the vertex removed from "set" to get "bestNeighbor"
  - vertex_added = the vertex added to "set" to get "bestNeighbor"

Note for part (c): For ***undergraduate*** students, this method may call `getNumEdges` for every neighbor of *set* to find the best neighbor. ***Graduate*** students: You may call `getNumEdges` ***only once*** at the start of this method. After that, each time you remove a vertex, *subtract* the number of edges that vertex contributes; when you a possible replacement, *add* the number of new edges the replacement contributes. (***Undergraduates:*** **+4** extra credit points for implementing this).

(d). `hillClimbClique(n, k, g, show_steps)` (20 points)

- Inputs:
  - n = the number of vertices in graph "g"
  - k = the number of vertices in list "set"
  - g = an undirected graph
  - show_steps = a boolean value that if true, prints at each the vertex eliminated and the vertex added

- Returns: list [finalSet, numEdges], where:
  - finalSet = list of k distinct vertices (in sorted order) for which no neighbor increases the number of pairs of adjacent vertices
  - numEdges = number of pairs of adjacent vertices in "finalSet"

- Starts by choosing a random set of k vertices and continues finding the best neighbor until the best neighbor does not increase the number of adjacent vertices.

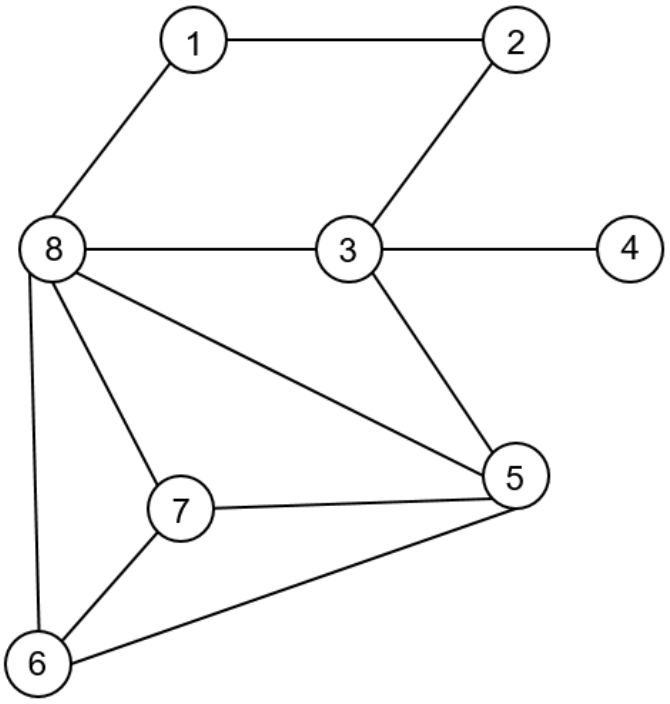
P2 (15 pts) Use hill climbing to find a clique of size 4 in the given graph (see later page), starting from various given start states (sets of 4 vertices). The new set at each step must be the ***BEST neighbor*** of the set in the previous step, i.e., the neighbor with the ***largest increase*** in the number of edges. For each new best neighbor, circle the new vertex that is added to the set.

P3 (20 pts) Use hill climbing to three-color the given graph (see later pages) starting from the given start states (3-colorings). At each step, give ***the BEST neighbor*** of the coloring in the previous step. In other words, the next step must give the ***largest decrease*** in the number of conflicts (edges with endpoints colored the same). For each step, indicate which vertex changed color as well as the number of conflicts there are. Also write the number of conflicts for the intial coloring.

For Problems 2 and 3, show your work on the sheets given, write your name on them, then scan them into a single PDF to submit along with your program from Problem 1 (and its output).

PB 2 **NAME:**_____

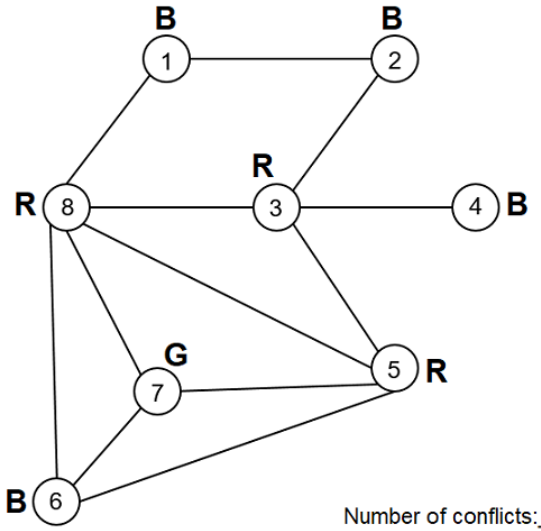
Use hill climbing to find a clique of size 4 in the graph below, starting from the start states given in the table below. For each new ***best*** neighbor, write how many pairs of adjacent vertices there are in the new set.



PB 2(a)				PB 2(b)				PB 2(c)		
	Set of size 4	Num Edges			Set of size 4	Num Edges			Set of size 4	Num Edges
START	**1, 2, 3, 4**	**3**		START	**3, 4, 5, 8**	**4**		START	**1, 2, 4, 6**	**1**
best neighbor				best neighbor				best neighbor		
best neighbor				best neighbor				best neighbor		
best neighbor				best neighbor				best neighbor		
best neighbor				best neighbor				best neighbor		
best neighbor				best neighbor				best neighbor		

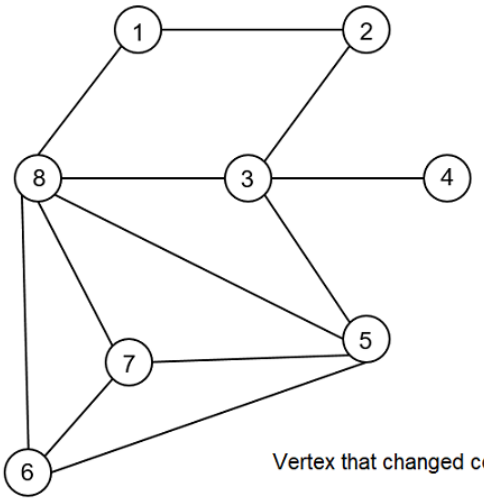
PB 3(a) **3 Colors: R, B, G** **NAME:**_____

START



Number of conflicts:_____

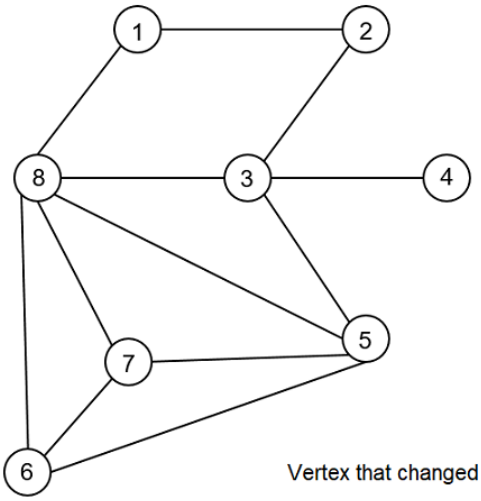
Step 3



Vertex that changed color: ____

New number of conflicts: ____

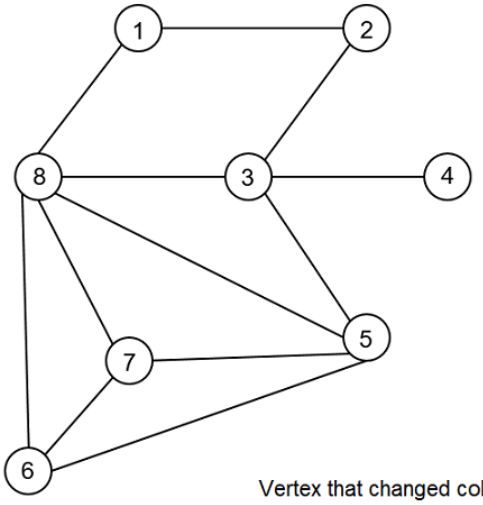
Step 1



Vertex that changed color: ____

New number of conflicts: ____

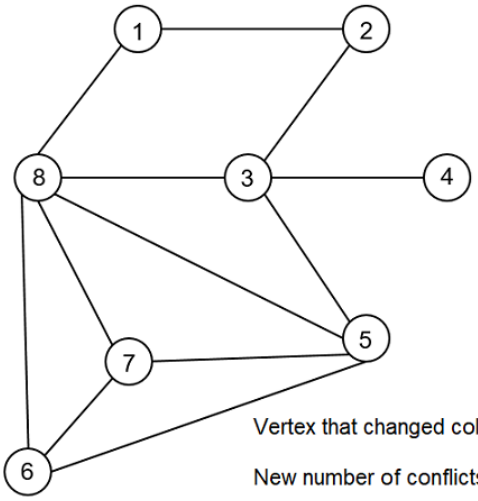
Step 4



Vertex that changed color: ____

New number of conflicts: ____

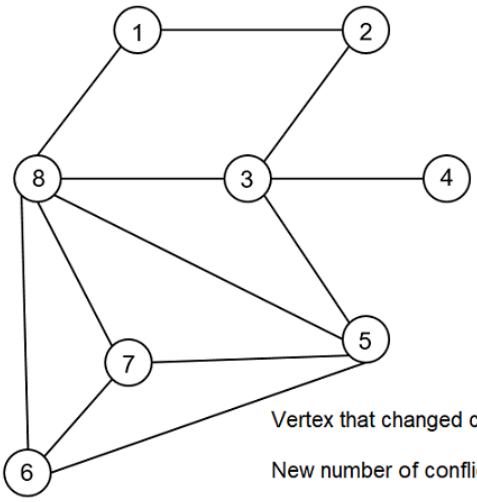
Step 2



Vertex that changed color: ____

New number of conflicts: ____

Step 5

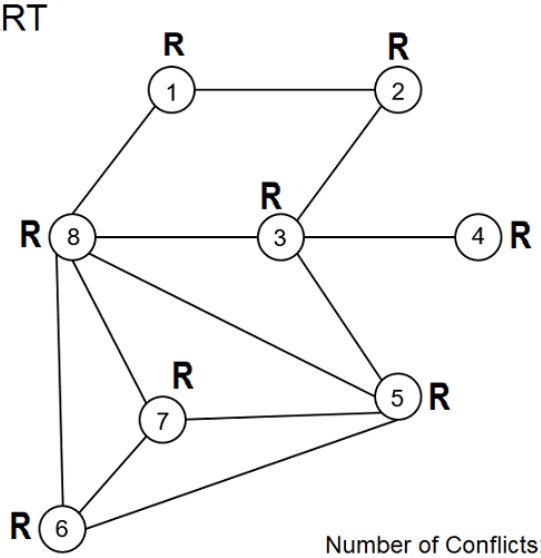


Vertex that changed color: ____

New number of conflicts: ____

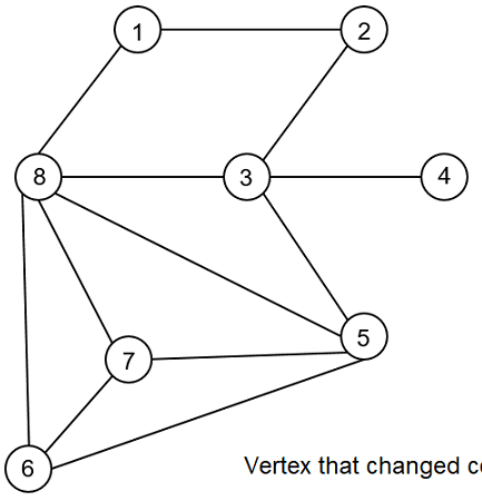
PB 3(b) **3 Colors: R, B, G** **NAME:**_____

START



Number of Conflicts: ____

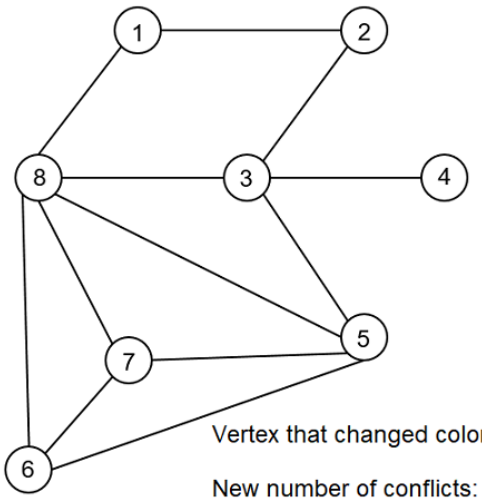
Step 1



Vertex that changed color: ____

New number of conflicts: ____

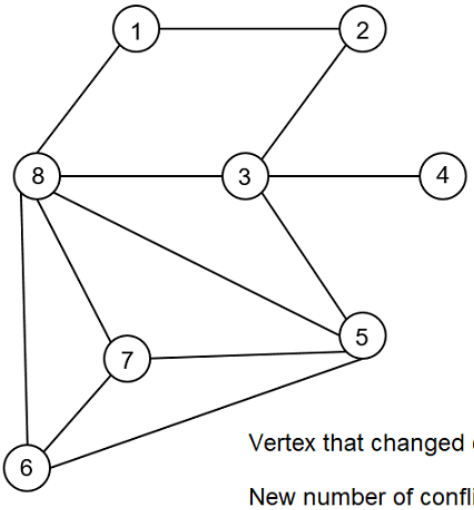
Step 2



Vertex that changed color: ____

New number of conflicts: ____

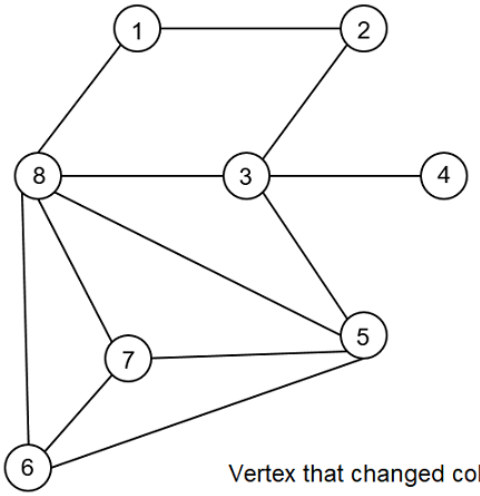
Step 3



Vertex that changed color: ____

New number of conflicts: ____

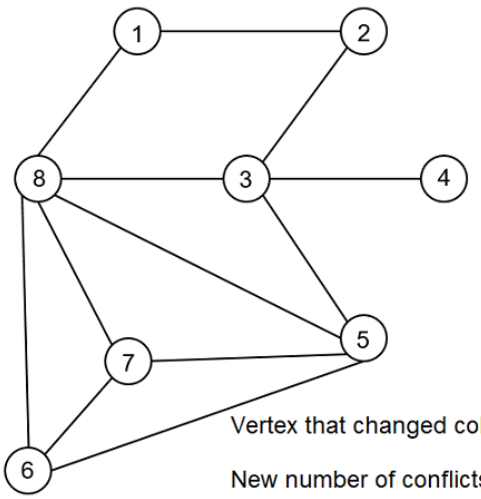
Step 4



Vertex that changed color: ____

New number of conflicts: ____

Step 5



Vertex that changed color: ____

New number of conflicts: ____