Question 1

A.

```
Node * curNode = m_head[v1], * prevNode;
for (;curNode && curNode -> ci != v2; curNode = curNode->m_next,prevNode =
curNode);
if (!curNode) {
    Node * newNode = new Node();
    newNode->ci = v2;
    if (prevNode)
        prevNode -> m_next = newNode;
    else
        m_head[v1] = newNode;
}
for (curNode = m_head[v2],prevNode=0;curNode && curNode -> ci != v1; curNode
= curNode->m_next,prevNode = curNode);
if (!curNode) {
    Node * newNode = new Node();
    newNode->ci = v1;
    if (prevNode)
        prevNode -> m_next = newNode;
    else
        m_head[v2] = newNode;
}
```

B.

CSR: O(n) + O(m) + O(m) = O(n) + O(2m) = O(n) + O(m)

AL: O(m) (Only as many nodes as there are edges)

CSR takes up more memory

C.

Insertion of an edge is faster in AL

CSR	
Find beginning of row using row extent	O(1)
Check to see if node already in array or find insert point	O(n)
Shift nonzero and column index arrays to add insert point	O(2m) = O(m)
O(1) + O(n) + O(m)	O(n) + O(m)

AL	
Find row	O(1)
Check to see if node is already in array	O(n)
If not, insert node	O(1)
O(1) + O(n) + O(1)	O(n)

If AL had to be in order, AL would still be faster because inserting a node would still be O(1) because all you need to do is change the m_next pointers, where as in CSR, you still need to shift the entire nonzero and column index arrays which is O(m)

D.

I'd prefer the AL over the CSR because implementing the AL is far simpler than implementing the CSR because inserting new nodes is as simple as creating a new node and then changing the pointers of old nodes. In addition, the AL is far easier to read and comprehend than the CSR method

Question 2

A.

```
Graph::matVec(int *x, int *y)
{
    for (int i = 0; i < m_numVert; i++)
    {
        y[i] = 0;
        for (int j = m_re[i], k = m_ci[j]; j < m_re[i + 1]; j++, k = m_ci[j])
            y[i] += m_nz[j] * x[k];
    }
}
```

B.

It wouldn't, because the original function already uses the column index to retrieve the corresponding value from x, so the order is irrelevant

Question 3

```
int i = 0;
for (k = A[(j = A[i])]; i < n && k < n; i++, k += n, k = A[(j = A[i] % n)]);
j;
```

- Loop through A (using I as an index)
- Add n to the value of A at ($j = A[i]\%\, n$)
    - $A[j]+= n$
- If we come across that number again, we'll know because A at that number will already be greater than or equal to n
    - $A[j] \geq n$

Question 4 Extra Credit

	1	2	3	4
1	1	2	5	10
2	2	2	5	10
3	5	5	5	10
4	10	10	10	10

Pair	Iternary	Travel Time	Total Travel Time
(1,2)	BWI->Princeton	2	**2**
2	Princeton->BWI	2	**4**
(3,4)	BWI->Princeton	10	**14**
1	Princeton->BWI	1	**15**
(1,2)	BWI->Princeton	2	**17**

1st. 1 and 2 go together, taking the standard train, 2 hours

2nd. 2 goes back with both tickets, taking the standard train, another 2 hours, 4 in total

3rd. 3 and 4 take both passes and go, taking the scenic route and stopping for that cheesesteak, another 10 hours, 14 in total

4th. 3 and 4 hand their passes to 1 who then goes back alone to pick up 2, just 1 hour, 15 in total

5th.Finally, 1 and 2 go to Princeton together, taking the standard train, another 2 hours, 17 in total