CISC 3150
Sample Final Exam

Question 1:

Trace the program and give the output for parts (a) through (d).

```
#include <iostream>
using namespace std;

class baseCL
{
public:
  baseCL(): dataBCL(5)
  {}

  void output()
  {    cout << dataBCL << "  "; }
protected:
  int dataBCL;
};

class derivedCL: public baseCL
{
public:
  derivedCL(int n = 1): dataDCL(n)
  { dataBCL = n + 1; }

  void output()
  {     baseCL::output();
    cout << dataDCL  << endl;

  }
private:
  int dataDCL;
};

int main()
{
  baseCL bObj;
  derivedCL d1Obj(8), d2Obj;


  bObj.output();         // part (a).     Output is _______
  cout << endl;

  d1Obj.output();        // part (b)      Output is _______
  d2Obj.output();        // part (c)      Output is _______

  bObj = d2Obj;
  bObj.output();         // part (d)      Output is _______
  cout << endl;

  return 0;
}
```

Question 2

This question refers to the following Java classes `D2` and `D3`:

```
class D2 {
    int x,y;
    public D2(int x, int y){
        this.x = x;
        this.y = y;
    }
    public boolean equals(Object o){
        if (!(o instanceof D2)){
            return false;
        } else {
            D2 d = (D2)o;
            return d.x == x && d.y == y;
        }
    }
}

class D3 extends D2 {
    int z;

    public D3(int x, int y, int z){
        super(x,y);
        this.z = z;
    }
    public boolean equals(Object o){
        if (!(o instanceof D3)){
            return false;
        } else {
            D3 d = (D3)o;
            return d.z == z && super.equals(o);
        }
    }
}
```

Give the output of each of the following code snippets:

2.1
```
D3 v1 = new D3(0,0,0);
D2 v2 = v1;
System.out.println(v1==(D3)v2);
```

2.2
```
D2 v1 = new D3(0,0,0);
D3 v2 = new D3(0,0,0);
System.out.println(v1==v2);
```

2.3
```
D2 v1 = new D2(0,0);
D3 v2 = new D3(0,0,0);
System.out.println(v1.equals(v2));
```

2.4
```
D2 v1 = new D2(0,0);
D3 v2 == new D3(0,0,0);
System.out.println(v2.equals(v1));
```

Question 3

The following gives a partial implementation of a class named `MyList` in C++ and Java. A `MyList` object is a singly-linked list, where the first node is referenced by the variable `head`, and the last node is referenced by the variable `tail`.

```
// Java
class ListNode<E> {
    public ListNode(E data, ListNode<E> next){
        this.data = data;
        this.next = next;
    }
    public E data;
    public ListNode<E> next;
}
class MyList<E> implements List<E> {
    ...
    private ListNode<E> head, tail;
}

// C++
template <typename E>
class ListNode {
public:
  E data;
  ListNode<E> *next;
  ListNode(E& item, ListNode<E> *ptr = NULL): data(item), next(ptr) {}
};

template <typename E>
class MyList {
  //    ...
private:
  ListNode<E> *head, *tail;
};
```

Implement the following methods (in C++ or Java) in the class `MyList`:

1. `equals(lst)`: This method returns `true` if `lst` equals this list; otherwise, it returns `false`.
2. `subList(fromIndex, toIndex)`: This method returns the portion of this list between the specified `fromIndex`, inclusive, and `toIndex`, exclusive.

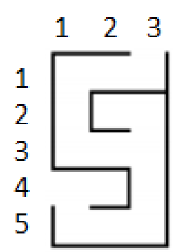
Question 4

The following function f takes $O(2^n)$ time to compute. Re-write the function using top-down dynamic programming to improve the efficiency.

```
int f(int n){
  if (n == 0) return 0;
  if (n == 1) return 1;
  return f(n-1) + 2*f(n-2);
}
```

Question 5

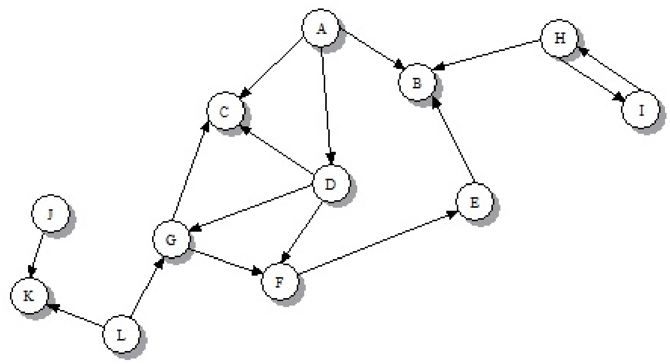
The function `solve_maze(Maze,R0,C0,R,C)` takes a maze, the position of a starting square (R0,C0), and the position of a target square (R,C), and returns a path from the starting square to the target square. The maze is given as a matrix, where each entry is a four-bit binary integer `(B3,B2,B1,B0)` that indicates how the corresponding square is connected to its neighboring squares: `B0` is 1 if the square is connected to the left; `B1` indicates if the square is connected to the right; `B2` indicates if the square is connected to the above; `B3` indicates if the square is connected to the below. For example, in the following maze, the square at (1,1) is represented by the binary number 1010 (i.e., the decimal number 10), meaning that the square is connected to the right and the blow.



A path is a list of visited square positions. Implement the function `solve_maze` in C++ or Java.

Question 6

This question refers to the functions `bfs` and `dfsVisit` defined on the separate pages, and the following graph:



6.1 List the vertices in reverse order of finish time for a depth-first visit of this digraph from vertex `A`.

`dfsList: _____ _____ _____ _____ _____ _____ _____`

6.2 List the set of vertices visited in a breadth-first search of the digraph from vertex `A`.

`Vertex set: _____ _____ _____ _____ _____ _____ _____`

6.3 What is the complexity of the depth-first search when a graph with V vertices and E edges is stored using the adjacency list representation?

```
set<T> bfs(graph<T>& g, const T& sVertex)
{
  queue<int> visitQueue;
  set<T> visitSet;
  int currVertex, neighborVertex;

  set<neighbor>::iterator adj;
  int i;

  currVertex = g.getvInfoIndex(sVertex);

  if (currVertex == -1)
    throw graphError("graph bfs(): vertex not in the graph");

  for (i=0;i < g.vInfo.size(); i++)
    if (g.vInfo[i].occupied)
      g.vInfo[i].color = vertexInfo<T>::WHITE;

  visitQueue.push(currVertex);

  while (!visitQueue.empty())
    {
      currVertex = visitQueue.front();
      visitQueue.pop();
      g.vInfo[currVertex].color = vertexInfo<T>::BLACK;

      visitSet.insert((*(g.vInfo[currVertex].vtxMapLoc)).first);

      set<neighbor>& edgeSet = g.vInfo[currVertex].edges;
      for (adj = edgeSet.begin(); adj != edgeSet.end(); adj++)
        {
          neighborVertex = (*adj).dest;

          if (g.vInfo[neighborVertex].color == vertexInfo<T>::WHITE)
            {
              g.vInfo[neighborVertex].color = vertexInfo<T>::GRAY;
              visitQueue.push(neighborVertex);
            }
        }
    }

  return visitSet;
}
```

```
void dfsVisit(graph<T>& g, const T& sVertex, list<T>& dfsList,
                     bool checkForCycle)
{
  int pos_sVertex, pos_neighbor;

  set<neighbor>::iterator adj;

  vector<vertexInfo<T> >& vlist = g.vInfo;

  pos_sVertex = g.getvInfoIndex(sVertex);

  if (pos_sVertex == -1)
    throw graphError("graph dfsVisit(): vertex not in the graph");

  vlist[pos_sVertex].color = vertexInfo<T>::GRAY;

  set<neighbor>& edgeSet = vlist[pos_sVertex].edges;

  for (adj = edgeSet.begin(); adj != edgeSet.end(); adj++)
    {
      pos_neighbor = (*adj).dest;
      if (vlist[pos_neighbor].color == vertexInfo<T>::WHITE)
        dfsVisit(g,(*(g.vInfo[pos_neighbor].vtxMapLoc)).first, dfsList, checkForCycle);
      else if (vlist[pos_neighbor].color == vertexInfo<T>::GRAY
               && checkForCycle)
        throw graphError("graph dfsVisit(): graph has a cycle");
    }

  vlist[pos_sVertex].color = vertexInfo<T>::BLACK;
  dfsList.push_front((*(g.vInfo[pos_sVertex].vtxMapLoc)).first);
}
```