Question 1:

```java
public static void DFS(Vertex a) {

    // Mark the current node as visited and print it
    a.isMarked();
    System.out.print(a.getID() + " ");

    // Visit adjacent vertices
    Iterator<Vertex> it = a.getNeighbours().keySet().iterator();
    while (it.hasNext()) {
        Vertex v = it.next();

        // Check if already visited
        if (!v.isMarked()) {
            DFS(v);
        }
    }

}
```

```java
public static void DFS(Vertex a) {

  // Mark the current node as visited and print it
  a.isMarked();
  System.out.print(a.getID() + " ");

  // Visit adjacent vertecies
  Iterator<Vertex> it = a.getNeighbours().keySet().iterator();
  while (it.hasNext()) {
       Vertex v = it.next();

       // Check if already visited before calling back
       if (!v.isMarked()) {
             DFS(v);
       }
  }

}
```

Question 2:

i)

Preferred data type for birthdays: String "DD-MM" e.g. "03-06", "31-12"

Because there is no need to have this data in separate variables, it just needs to be unique for each different birthday and stored in a single value for easy comparison

ii)

Using HashSet so there is only one iteration instead of nested for loops which would be O(n^2)

```
// returns 1 when there is a match and 0 for no match
int atleastTwoSame(String bday1, String bday2, HashSet<String, Integer> bdayAndCount) {
      for (String name : names) {
         Integer count = bdayAndCount.get(name);
         if (count == null) {
            bdayAndCount.put(name, 1);
         } else {
            bdayAndCount.put(name, ++count);
         }
      }
      // Print duplicate element
      Set<Entry<String, Integer>> entrySet = bdayAndCount.entrySet();
      for (Entry<String, Integer> entry : entrySet) {
         if (entry.getValue() > 1) {
           System.out.printf("duplicate element '%s' with count '%d' :", entry.getKey(),
entry.getValue());
            return 1;
         }
      }
return 0;
}
```

iii)

O(n) because this is linear worst case is all birthdays are compared in one iteration of the loop which is n operations