Course of Programming Paradigms

Prof. Vittorio Maniezzo

Notice:

- The following specifications will be checked automatically. And it's fundamental to respect them in detail, with particular reference to the names proposed. Otherwise the project will probably not work.

- The failure of the implementation of some specifics, will result in a corresponding reduction of the final evaluation, up to a level required to repeat the test
examination.

It is required to build an application of type Class Library named ASDlib (in a single fileASDlib.cs) and containing the namespace ASDlib.
The library exposes the following interfaces:

```
interface IPriorityQueue
{  bool insert(int i);
   int findMin();
   int extractMin();
}
interface IGraphSearch
{  void depthFirst();
   void breadthFirst(int s);
}
interface ICandidate
{  string nome {get;}
   string cognome {get;}
   string matricola {get;}
}
```

Internally manages a graph, stored as:

```
public struct Nodo
{  public int id;
   public int x,y;
   public Nodo(int p1, int p2, int p3){id=p1;x=p2;y=p3;}
}
public struct Arco
{  public int id;
   public int end1;
   public int end2;
   public int w;
   public Arco(int p1, int p2, int p3, int p4)
   {  id = p1; end1 = p2; end2 = p3; w=p4;
   }
}
```

Are required:

- A **public class Ordinamenti** with methods overloading for int, double, and string for insertionSort, QuickSort (signature: public void insertionSort (int [] A), public void quicksort (int [] arr) and overloading.
  Implement also countingSort (public void countingSort (int [] A, int out[]B)only for arrays
  integers.
  Non-recursive implementations are preferred to avoid stack overflow, but the client will not check for this. Consider also the possibility of defining in overriding the operator < for the
  base class string (i will not check, but can you? how?).

- A **public class Graph**, the graph represented as List <Nodo>nodi an List<Arco>archi.
  The class exposes a method *public void readXMLgraph (string fpath)* and the virtual methods *Kruskal (public virtual List<int> Kruskal()), Prim(public int[]Prim (int r)) and Dijkstra (public int[] Dijkstra(int s))*, also public variables *numNodi*, *numArchi* and boolean *isOriented*.
  The class graph must then be specialized in two derived classes, **GrafoOrientato** and **GrafoNonOrientato** that redefine (only where it makes sense) in overriding the base class methods, returning *null* or an array of int with predecessors (Dijkstra, Prim) or a List <int> (Kuskal).

  Note: Kruskal inside will make use of an instance of the **public class UpTree** *(public constructor UpTree (int n))* which exhibits the *findSet (public int findSet (int x))*, *makeSet (public void makeSet (int x))* and *union (public void union (int x, int y))*.

- A **public class MyHeap** that exposes properties *HeapInt*, *HeapDouble* and *HeapString* **only read**, linked to similar private arrays, and public methods *buildHeap (public void*
  buildHeap (int [] A)), insert (public void Insert (int x)), extractMin (public void extractMin(out int min)), with overloading for integers, doubles and strings. Where necessary, it is required to maintain a resizing of the initial array.
  The methods buildHeap, insert and extractMin work with HeapInt, and HeapDouble HeapString.

  NOTE: The heap must order from smallest to largest!

- A **public class MyHash** that exposes **read only properties** NumPos, connected to a
  private var. m **itialized by constructor**. Hash function for division (k% m) in private method. The data is unstructured (data are also its key) and correspond to whole numbers.
  Public Methods: *List <int> showTableLine (int k)* that returns the list corresponding to the position k of the table. *chainedHashInsert (int x)*, *bool chainedHashSearch (int k)* and
  bool chainedHashDelete(int x).

The interface **IPriorityQueue** must be realized either by a class **ArrayPQ**, based on a private array initialized by the constructor (no resize), an by a **HeapPQ** that implement the priority queue respectively with an array and with an object of type MyHeap.

The interface **IGraphSearch** must be realized by a class GraphSearch which works on an private object type Graph (either directly or following upcast from derived classes) contained in files called "Grafo.xml", which can be oriented or not, and exposes three arrays of integers *p*, *d* and *f*, and an array *nColor []nodeColor* , where nColor is an enum defined on the set of values {white, gray, black}

The interface **ICandidate** must be implemented from class Ordinances.