```
/*
 * =====================================================================================
 *
 *       Filename:  nodes.h
 *
 *    Description:  description of phylogentic nodes as per
 *    Fitch and Sannkoff as describd by Felenstein
 *
 *        Version:  1.0
 *       Created: Nov  4 21:15:49 EDT 2016
 *       Revision:  none
 *       Compiler:  gcc
 *
 *         Author:  Ruben Safir,
 *        Company:  LIU Thesis
 *
 * =====================================================================================
 */
#ifndef NODES_H
#define NODES_H
#include<iostream>
#include<string>
#include<vector>
namespace tree {


/* ================================================================================
 * Class NODE -
 * Description - This is a node in the tree that must undergo parsimony evaluation
 * ================================================================================
 */
template<class unk>
class NODE{
   public:
      //=========================LIFECYCLE=============================================
      /* Constructor */
      NODE<unk>( unk states, NODE<unk> *cl= 0, NODE<unk> *cr = 0, NODE<unk> *p = 0 )
;
      ~NODE<unk>();

      /* ====================  ACCESSORS     ======================================
 */
      inline unk states()const{
         return _states;
      };
      inline NODE<unk> * cl()const{
         return _cl;
      };
      inline NODE<unk> * cr()const{
         return _cr;
      };
      inline NODE<unk> * p()const{
         return _p;
      };

      inline void states(unk phenotype){
         _states = phenotype;
      }
      inline void cl(unk *child){
         _cl = *child;
      }
      inline void cr(unk *child){
         _cr = *child;
      }
      inline void p(unk *parent){
         _p = *parent;
      }
      /* ====================  MUTATORS      ======================================
 */

      /* ====================  OPERATORS     ======================================
```

```
 */

   protected:
      /* ====================  DATA MEMBERS  ======================================
 */


   private:
      /* ====================  DATA MEMBERS  ======================================
 */
      unk _states;
      NODE<unk> _cl, _cr, _p;


}; /* -----  end of template class NODE  ----- */


/*
 * ===========================================================================
====
 *        Class:  State
 *  Description:  This describes a possible individual state or charactoristic
 * ===========================================================================
====
 */
template<class cost>
class state
{
   public:
      /* ====================  LIFECYCLE     ======================================
 */
                              /* constructor */
      explicit inline state<cost>(std::string const a, cost r)//r is the current min
imal state for a node
      :_nam{a}, _r{r}
      {};


      /* ====================  ACCESSORS     ======================================
 */
      inline std::string nam(){return _nam;};
      inline cost r(){return _r;};
      //once it is set it is set
      //inline void nam(std::string a ){_nam = a;}
      inline void r(cost b ){_r = b;}
      /* ====================  MUTATORS      ======================================
 */

      /* ====================  OPERATORS     ======================================
 */

      /* ====================  DATA MEMBERS  ======================================
 */
   protected:

   private:
      std::string _nam;
      cost _r;

}; /* -----  end of class State  ----- */

/*
 * ===========================================================================
====
 *        Class:  Pstates
 *  Description:  vector of all possible states
 * ===========================================================================
====
 */
template<class cost>
class Pstates
{
   public:
```

```
      /* ====================  LIFECYCLE     ======================================
 */
      inline explicit Pstates (std::vector< state<cost>  > x)
         :_vstate{x}{};                              /* constructor      */
      inline ~Pstates (){delete _vstate; };                            /* destructor
       */

      /* ====================  ACCESSORS     ======================================
 */

      /* ====================  MUTATORS      ======================================
 */

      /* ====================  OPERATORS     ======================================
 */

      const Pstates& operator = ( const Pstates &other ); /* assignment operator */

      /* ====================  DATA MEMBERS  ======================================
 */
   protected:

   private:
      std::vector<state <cost> > _vstate;

}; /* -----  end of class Pstates  ----- */

/*
template<class unk>
inline unk NODE<unk>::value(){ if(this != NULL) return _value; else return NULL; };

template<class unk>
inline void NODE<unk>::value(unk val){
   _value = val;
};

*/


/*
template<class unk>
class LIST{
   public:
      LIST<unk>() : _at_front(0), _at_end(0), _size(0) {}
      inline int size();
      inline void insert(NODE<unk> *, unk);
      inline void insert(unk);
      inline void front(unk);
      inline void up_size();
      inline void down_size();
      void display(ostream &os = cout);
      void remove_front();
      void remove_all();
      void remove_item(unk);
      void remove_item(NODE<unk> *);
      NODE<unk> * find(unk);
      NODE<unk> * find_all(unk, NODE<unk> * last = NULL );

      ~LIST<unk>();

   private:
      NODE<unk> *_at_front;
      NODE<unk> *_at_end;
      int _size;
      LIST<unk>(const LIST<unk>&);
      LIST<unk>& operator=( const LIST<unk>&);
};
*/

/*
template<class unk>
void LIST<unk>::display(ostream &os){
```

```
   NODE<unk> * tmp = _at_front;

   if (tmp == 0){
      os << "No List" << endl;
      return;
   }


   //unk i =0;
   while(tmp != _at_end){
      //cout << "Entering While Loop: "<< ++i << endl;
      os << tmp->value() << ":";
      tmp = tmp->next();
   }

   os << tmp->value() << endl;

}
*/
}        /*---End of Namespace TREE---*/
#endif
```