```
#include <string>
#include <iostream>
#include <iomanip>


// A new thin wrapper around a string of characters
template<typename charT, typename traits = std::char_traits<charT> >
class center_helper {
  std::basic_string<charT, traits> str_;
  public:
    center_helper(std::basic_string<charT, traits> str) : str_(str) {}

    template<typename a, typename b>
    friend std::basic_ostream<a, b>& operator<<(std::basic_ostream<a, b>& s,
        const center_helper<a, b>& c);
};

// Allows us to center things that are convertable to a string
template<typename charT, typename traits = std::char_traits<charT> >
center_helper<charT, traits> centered(std::basic_string<charT, traits> str) {
  return center_helper<charT, traits>(str);
}

// Allows us to center strings
center_helper<std::string::value_type, std::string::traits_type> centered(
    const std::string& str) {
  return center_helper<std::string::value_type, std::string::traits_type>(str);
}

// Does the centering
template<typename charT, typename traits>
std::basic_ostream<charT, traits>& operator<<(std::basic_ostream<charT, traits>& s,
    const center_helper<charT, traits>& c) {

  std::streamsize w = s.width();
  if (w > (int)c.str_.length()) {
    std::streamsize left = (w + c.str_.length()) / 2;
    s.width(left);
    s << c.str_;
    s.width(w - left);
    s << "";
  } else {
    s << c.str_;
  }
  return s;
}
```