Quin'darius Lyles-Woods

-Data Structures 3305

-Assignment 3

-Sherry Perry

Code

```
#include "set.h"
#include <iostream>
#include <cassert>



 set::set(size_type initial_capacity) {
    capacity = initial_capacity;
    used=0;
    data=new value_type[capacity];

}

inline set::~set()
{
    capacity=0;
    used =0;
    delete data;
}

 set::set(const set& s)
{     capacity=s.capacity;
```

```
    used=s.used;
    data= new value_type[capacity];

    for (int i=0;i<used;i++)
    {
        data[i] = s.data[i];
    }


}

 bool set::erase(const value_type& target)
{
    bool in=false;
    int init=0;
    for(int x=0;x<capacity;x++)
    {
        if(target==data[x])
        {
            in=true;
            break;
            init=x;
        }
    }
    if(in==true)
    {
        for(int y=init;y<capacity-1;y++)
        {
            data[y]=data[y+1];
        }
    }
    return in;

}
 bool set::insert(const value_type& entry)
{
    bool found = false;
    for (int i = 0; i < used; i++)
        if (data[i] == entry)
        {
            found = true;
            break;
        }
    if (!found && used < capacity)
    {
        data[used] = entry;
        used++;
    }
    return found;
}
 set& set::operator =(const set& s)
{
    set r(s.capacity);
    r.used=s.used;
    data= new value_type[capacity];

    for (int i=0;i<used;i++)
    {
        data[i] = s.data[i];
    }
    return r;
}
 set::size_type set::size()const
{
    return used;
}
 bool set::contains(const value_type& target)const
{
    bool con =false;
    for(int x=0;x<capacity;x++)
    {
        if(target==data[x])
        {
            con=true;
        }
    }
    return con;
}
```

```
 std::ostream& operator << (std::ostream& output, const set& s)
{
    output << "{";
    for(int i=0;i<s.size()-1;i++)
        output << s.data[i] << ",";
    output << s.data[s.size()-1] << "}";
    return output;
}
```