```
Script started on Fri 01 Apr 2016 12:55:50 AM CDT
\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ pwd
/home/students/g_butler4/csc122/dataval
\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ cat info.txt
Gary Butler
CSC122-002
Project:Data=Value(lab)
Levels Attempted:7
Description:This program reads xml tagged data and stores it to be written to another
file.\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ cat gettype.h
#ifndef GETTYPE_H
#define GETTYPE_H

#include <string>

int get_type(std::string in,std::string (&cat)[4])
{
        char type;
        bool f = false;

        int t = (in.find_first_of('<', 0) + 1);
        type = in[t];
        if ((type == 's'))//<student>
        {
                return 48;
        }
        if (type == '/')//</student>
        {
                return 50;
        }
        for (int i = 0; i < 3; i++)
        {
                if (type == cat[i][0])//check in to wordset
                {
                        type = in[t + 1];
                        if (type == cat[i][1])
                        {
                                f = true;
                                return (i + 1);
                        }
                        else if (type == cat[i + 1][1])
                        {
                                f = true;
                                return (i + 2);
                        }

                }
        }
        if (f == false)//word not found
        {
                return 49;
        }

}

#endif\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ cat in.txt
<student>
    <name>Jason James</name>
    <id>123456</id>
    <gpa>9.2</gpa>
    <grade>B</grade>
    <gender>m</gender>
</student>
<student>
    <name>Tammy James</name>
    <gpa>11.2</gpa>
    <grade>A</grade>
    <id>123457</id>
    <gender>f</gender>
</student>
<student>
    <name>Henry Ramirez</name>
    <gpa>12.3</gpa>
    <id>111888</id>
    <major>ChE</major>
    <class>soph</class>
    <gender>m</gender>
</student>
<student>
    <id>788531</id>
    <name>Suzie Shah</name>
    <grade>Q</grade>
</student>
\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ cat dv.cpp
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
#include "gettype.h"

using namespace std;


class info{
        string name;
        string id;
        string gpa;
        string gen;

public:


        void add_name(string in){ name = in; }
        void add_id(string in){ id = in; }
        void add_gpa(string in){ gpa = in; }
        void add_gen(string in){ gen = in; }

        string get_name(){ return name; }
        string get_id(){ return id; }
        string get_gpa(){ return gpa; }
        string get_gen(){ return gen; }


        void reset()
        {
                name = "0";
                id = "0";
                gpa = "0";
                gen = "0";
        }

        info()
                :name("0"), id("0"), gpa("0"), gen("0")
        {}
};

string cat[4] = { "name", "id", "gpa", "gender" };
```

```
int main()
{
        string fn;
        string temp;
        fstream filei;
        ofstream fileo;


        vector<info> list;
        vector<info>::iterator i;
        info temp_o;


        cout << "Enter name of file: ";
        getline(cin >> ws, fn);
        if (fn.find_first_of('.') == -1)
        {
                fn.append(".txt");
        }
        filei.open(fn.c_str());


        if (filei.fail())
        {
                while (filei.fail())
                {
                        filei.close();
                        filei.clear();
                        cout << "\nTry again";
                        cout << "\nEnter name of file: ";
                        getline(cin >> ws, fn);
                        if (fn.find_first_of('.') == -1)
                        {
                                fn.append(".txt");
                        }

                        filei.open(fn.c_str());
                }
        }
        else
        {
                cout << "Opening: " << fn<<"\n";
        }
        filei.peek();
        while (!(filei.eof()))
        {
                getline(filei, temp);

                while (!temp.empty())
                {
                        switch (get_type(temp, cat))
                        {
                        case 1://name
                        {
                                                temp.erase(0, (temp.find_first_of('>', 0) +
 1));
                                                temp.erase(temp.find_first_of('<', 0), temp
.find_first_of('>', 0));
                                                temp_o.add_name(temp);
                                                temp.clear();
                        } break;
                        case 2://id
                        {
                                                temp.erase(0, (temp.find_first_of('>', 0) +
 1));
                                                temp.erase(temp.find_first_of('<', 0), temp
.find_first_of('>', 0));
                                                temp_o.add_id(temp);
                                                temp.clear();
                        }break;
                        case 3://gpa
                        {
                                                temp.erase(0, (temp.find_first_of('>', 0) +
 1));
                                                temp.erase(temp.find_first_of('<', 0), temp
.find_first_of('>', 0));
                                                temp_o.add_gpa(temp);
                                                temp.clear();
                        }break;
                        case 4://gender
                        {
                                                temp.erase(0, (temp.find_first_of('>', 0) +
 1));
                                                temp.erase(temp.find_first_of('<', 0), temp
.find_first_of('>', 0));
                                                temp_o.add_gen(temp);
                                                temp.clear();
                        }break;
                        case 50://end of student, flush vars
                        {
                                                        temp.clear();
                                                        list.push_back(temp_o);
                                                        temp_o.reset();
                        }break;

                        default:
                        {
                                                temp.clear();
                        }

                        }break;
                        filei.peek();

                }
        }
        filei.close();

        cout << "Enter file to be written to: ";
        cin >> fn;
        if (fn.find_first_of('.') == -1)
        {
                fn.append(".txt");
        }
        if (isalpha(fn[0]) || isdigit(fn[0]))
        {
                cout << "\nwriting to: " << fn;
        }
        else
        {
                while ((!isalpha(fn[0])) && (!isdigit(fn[0])))
                {
                        fn.clear();
                        cout << "\nTry again";
                        cout << "\nEnter file to be written to: ";
                        cin >> fn;
                        if (fn.find_first_of('.') == -1)
                        {
                                fn.append(".txt");
                                }
                        }
                }
                fileo.open(fn.c_str());

                        for ((i = list.begin()); (i != list.end()); (i++))
                        {
                                fileo << "<student>\n";
                                if ((i->get_name()) != "0")
                                {
                                        fileo << "<name>" << i->get_name() << "</name>\n";
                                }
                                if ((i->get_gpa()) != "0")
                                {
                                        fileo << "<gpa>" << i->get_gpa() << "</gpa>\n";
                                }
                                if ((i->get_id()) != "0")
                                {
                                        fileo << "<id>" << i->get_id() << "</id>\n";
                                }
                                if ((i->get_gen()) != "0")
                                {
                                        fileo << "<gender>" << i->get_gen() << "</gender>\n";
                                }
                                fileo << "</student>"<<"\n";
                        }
                        fileo.close();

                return 0;
}

\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ CPP gettype.h dv.c
pp
dv.cpp***
In file included from dv.cpp:5:
gettype.h: In function ‘int get_type(std::string, std::string (&)[4])’:
gettype.h:44: warning: control reaches end of non-void function
In file included from dv.cpp:5:
gettype.h:46:7: warning: no newline at end of file
dv.cpp:53:   instantiated from here
dv.cpp:53:   instantiated from here
dv.cpp:53:   instantiated from here
dv.cpp:54:   instantiated from here
dv.cpp: In function ‘int main()’:
dv.cpp:60: warning: comparison between signed and unsigned integer
expressions
dv.cpp:76: warning: comparison between signed and unsigned integer
expressions
dv.cpp:99:   instantiated from here
dv.cpp: In function ‘int main()’:
dv.cpp:146: warning: comparison between signed and unsigned integer
expressions
dv.cpp:162: warning: comparison between signed and unsigned integer
expressions
dv.cpp:138: warning: will never be executed
dv.cpp:128:   instantiated from here



\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ CPP gettype.h dv.c
pp\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[K\033[
K\033[K\033[K\033[K\007\007\007\007\007\007\007\007\007\007\007\007./dv.out
Enter name of file: in.txt
Opening: in.txt
Enter file to be written to: out

writing to: out.txt\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$
 cat out.txt
<student>
<name>Jason James</name>
<gpa>9.2</gpa>
<id>123456</id>
<gender>m</gender>
</student>
<student>
<name>Tammy James</name>
<gpa>11.2</gpa>
<id>123457</id>
<gender>f</gender>
</student>
<student>
<name>Henry Ramirez</name>
<gpa>12.3</gpa>
<id>111888</id>
<gender>m</gender>
</student>
<student>
<name>Suzie Shah</name>
<id>788531</id>
</student>
\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ \033[3P./dv.out
Enter name of file: name
Opening: in.txt
Enter file to be written to: out

writing to: out.txt\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$
 cat out.txt
<student>
<name>Jason James</name>
<gpa>9.2</gpa>
<id>123456</id>
<gender>m</gender>
</student>
<student>
<name>Tammy James</name>
<gpa>11.2</gpa>
<id>123457</id>
<gender>f</gender>
</student>
<student>
<name>Henry Ramirez</name>
<gpa>12.3</gpa>
<id>111888</id>
<gender>m</gender>
</student>
<student>
<name>Suzie Shah</name>
<id>788531</id>
</student>
\033]0;g_butler4@mars:~/csc122/dataval\007[g_butler4@mars dataval]$ cat tpq.txt
1.My main handles the opening and closing of the file, as well as the reading and writ
ing.
2.My code sets the fstream to read the file line by line.
3.I compared the xml code of the read line to match given keywords that are stored in
a global array.
4.Xml format would ignore any comments.
5.All data lines should be labled on input or they are ignored.
6.If the line does not contain <> to start or end with, it is ignored.
7.The code reads the beginning tag, the data, then the closing tag. Once the closing t
ag is found and appended, the line is cleared.
8.My code would lose the current object’s incoming data, but preserve sucessfully read
 objects.
9.Default values are set to "0", and are ignored in the write process. If they are not
 the default value, they are written to the new file.
10.I search for the first bracket, then find the next availible bracket and erase it.
From there, the data type is determined and its contents are extracted into an object.
11.No one should have a bracket in their name, but if they did my code would read all
data before the bracket and erase the rest. Otherwise, all data would be stored in my
object including a student named "=".
12.Spacing around the lable is ignored, however spacing within the data would result i
n the preservation of that space in the written file.
13.I compare data to the before mentioned array of lables. If the tag does not match t
he lable, it is discarded.
14.No, my code will not handle infomation on multiple lines nor will it handle multipl
e items on one line. My code is set to only read one line as one peice of possible dat
a.
15.Since the data is not being accessed, it doesnt matter what data type the file is s
tored as. All data types in my code are turned to strings for ease of use.
16.A string can be a bool if set to two conditions (M/F) and a string to char conversi
on is built in.
17.Because they only contain functions.\033]0;g_butler4@mars:~/csc122/dataval\007[g_bu
tler4@mars dataval]$ exit
exit

Script done on Fri 01 Apr 2016 12:58:55 AM CDT
```