Goals:

1. Learn how to use **Exception**.
2. Learn how to use **getline() function** to read data from a file.
3. Learn how to write **robust** code.

Requirement:

1. The input file named students.txt consists of four columns.

	Column name	Definition
1	Action	'I' – Insert the record. 'U' – Update the record with the new data. (SSN is the key.) 'D' – Delete the record which has the same SSN.
2	SSN	Social security number
3	Name	The data is delimited by double quotes. There might be extra blanks in the data.
4	Sex	It is either male ("M") or female ("F ").

2. The input data may contain exceptions which should be handled by using **try/catch/throw.** There are at two types of exceptions. **First type of exceptions** is invalid SSN. For example, the SSN 121219ooo in the following input file is not digit. **Second type of exceptions** is logic errors. The last three lines in the input file cause this type of exceptions. Whenever there is an exception, the catch statements in the system should prompt a proper exception message in which the social security number is included.

I 123456789 "John J. Johnson" F I 512434990 " Mary Jackson" F I 342432444 "Peter Von Young" M I 470068625 "Jim Lee" M I 234324324 T "Tammy Gaddis" F I 121219000 "Ester Schwab " F U 123456789 "John Johnson" M D 121219ooo " Ester Schwab" F D 121219000 " Ester Schwab" F	<- Error prompt, SSN is not an integer

D 121219000 " Ester Schwab" F I 123456789 "John Johnson" F U 999999999 "John Trumpt " M	<- Error prompt, no such record <- Error prompt , record exists <- Error prompt, no such record

3. Because a name may consist of many parts, the **getline()** function has to be used to read the entire line at a time. The line will have to be decomposed into proper data tokens. Since extra blanks may appear anywhere in the input file, the functions for processing the data tokens should be as **robust** as possible.
4. After the data are loaded in the data structure, the user may enter the queries, either 'a' or 's,' or enter 'q' to quit the operation.

a	Display all student information
s	The system will ask the user to enter the SSN of the student. With the SSN the system will return the information of the student. If the SSN does not exist, an exception is prompted.
q	Quit

5. (10%) On the top of the main.cpp file explain why statements:

```
cin.clear();
cin.ignore(32768, '\n');
```

are used.

Skeleton code

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

using namespace std;

struct Student
{
        int ssn;
```

```
        string name;
        char gender;
};

class StudentMgr
{
private:
        vector<Student> sList;

        void insertStudent(int, string, char);
        void updateStudent(int, string, char);
        void deleteStudent(int);

        int findStudentIndex(int); //binary search using SSN
        void sortBySSN();

public:
        StudentMgr();
        ~StudentMgr();

        void displayStudent(int);
        void displayAll();
};

StudentMgr::StudentMgr()
{
   /// TO DO: Add your code here
   ///
   /// In each iteration of a while loop do the following:
   ///      1. Use getline() to read one line of data in students.txt
   ///      2. Use the code in parseString() to decompose the data tokens.
   ///         And save the data tokens in a local vector. (This is the vector sList.)
   ///      3. Conduct an action. An action is one the following
   ///              'I' - Insert the record. Use push_back to add the record to the end
   ///               of the vector sList.
   ///              'U' - Update the record
   ///              'D' - Delete the record
```

```
   ///
   ///  Sort the vector after all records are added.
}

StudentMgr::~StudentMgr()
{

}

void StudentMgr::insertStudent (int ssn, string ame, char gender)
{
   /// TO DO: Add your code here
}

void StudentMgr::updateStudent (int ssn, string ame, char gender)
{
   /// TO DO: Add your code here
}

void StudentMgr::deleteStudent (int ssn)
{
   /// TO DO: Add your code here
}

void StudentMgr::displayStudent(int ssn)
{
   /// TO DO: Add your code here
}

void StudentMgr::displayAll()
{
   /// TO DO: Add your code here
}

int StudentMgr::findStudentIndex( int ssn)
{
   /// TO DO: Add your code here
```

```
   /// Binary search should be conducted
   /// If the record cannot be found, throw an integer -1 here to signal that
   /// an exception has occurred.
}

void StudentMgr::sortBySSN()
{
   /// TO DO: Add your code here
   /// You may or may not use this function. You may append new record to the
end of the
   /// vector. After all records are appended, sort all of them at a time.
}


int main() {
	StudentMgr sm;
	char input;
	while (input!='q') {
		cout << endl << endl << "Select an option:" << endl;
		cout << "-----------------" << endl << endl;

		cout << "A. Display all student info" << endl;
		cout << "S. Display specific student" << endl;
		cout << "Q. Quit the program" << endl;
		cout << "-------------------------------" << endl;
		cout << "Enter selection: ";
		cin >> input;
		cin.clear();
		cin.ignore(32768, '\n');
		input = tolower(input);

		switch (input) {
		case 'a':
			sm.displayAll();
			break;
		case 's':{
			cout << "Input SSN: ";
```

```
                        int ssn;
                        cin >> ssn;
                        cin.clear();
                        cin.ignore(32768,'\n');
                        sm.displayStudent(ssn);
        }
                }

        }

        return 0;
}
```