CS141: Intro to Computer Science I
Streams and Character I/O

Objectives:

Understanding File I/O and Character I/O

Streams:

A stream is a communication channel where one entity can place data while another reads it. We have seen two of these streams already. Cout is a call to the standard output stream. Data from the program is placed into it, while the console reads out of it. Cin is a call to the standard input stream. Data from the console is placed into the stream and the program accepts it. We can also create a stream between files.

In file input, the contents of the file are placed in the stream and the program can read them. In file output the reverse happens. The program puts its output into the stream, which gets placed into the file, instead of going to the console.

File I/O:

Most of what we've done up to this point has only been temporary. Everything our program does is printed to the screen. Once the terminal is closed, we lose any results. File output allows us to save results or data to a file. File input allows us to use stored data to put a system into a state we left last time.

How to use File I/O:

A. Include the file stream

I. #include<fstream> This includes file streams in your program.

B. Declare the file stream

I. ofstream streamName; This declares an output file stream, with the name "streamName". This is used for output only.

II. ifstream streamName;This declares an input file stream, with the name "streamName". This is used for input only.

III. fstream streamName;This declars a file stream with the name "streamName". This is used for input and output.

C. Open File

I. streamName.open("file.txt");

This would connect the file “file.txt” with the program via the stream “streamName”.
After opening a file, you should ensure it opened correctly. You can use the command

streamName.is_open() - Returns true if file is open, false otherwise

streamName.fail() - Returns true if fail bit is set (indicating opening failed)

D. Using Stream

I. You can use an output stream just like you use cout

```
ofstream out;
out.open(“file.txt”);
if(!out.fail())
{
        out << “This would end up in the file”;
}
```

II. You can use an input stream just like you would cin

```
ifstream in;
in.open(“file.txt”);
 if(!in.fail())
{
        int x;
        in >> x;
}
```

E. Closing the stream

When you are done using a stream, you should always remember to close it before exiting the program. To close a stream, simply use the close command.

```
streamName.close()
```

F. Useful functions and flags

I. streamName.eof()

Returns true if we are at the end of a file, useful for looping through files.

II. streamName.fail()

Returns true if the stream has the fail bit set. This is especially useful when opening files,

if a file fails to open, the fail bit will be set.

Character I/O:

Sometimes we don't want to read an entire string, or an entire number at one time. Using ">>" only allows us to read until we find a space or the end of the type. What if we only wanted the first digit of a number? Or the first letter of a string? Also consider the case where we don't know what type of data we're looking at. If we try to use ">>" to read an integer, but see a character instead, it will cause an infinite loop. Can we look ahead to make sure we're reading the right thing?

Once we have a filestream open, we can perform some operations on it. (Note: all of these operations can be done using cin too). For these examples consider we have a ifstream named "in" and a ofstream "out" already opened.

A. Common Commands

I. in.get() This gets the next character in the stream, removes it from the stream, and returns it.

Example:

```
char c;
c = in.get()
```

II. in.peek() This will get the next character in the stream. Unlike get, it does not remove the character, it lets us look ahead. It returns the first character in the stream.

Example:

```
char c;
c = in.peek();
```

III. in.putback(char) This places the character "char" back at the beginning of the stream, it is the opposite of a get(). There is a similar function, unget(char).

Example:

```
char c = in.get()
in.putback(c)
```

IV. out.put(char) This places the character at the end of the stream,

Example:

```
char c = in.get()
 out.put(c)
```

V. There are more functions, but these will cover most of what you need for character I/O

B. Helper functions for character I/O

I. isdigit(char) This function returns true if the character “char” is a digit [0-9]

II. isalpha(char) This function returns true if the character “char” is a letter [a-z or A-Z]

III. isupper(char) This function returns true if the character “char” is an upper case letter [A-Z]

IV.islower(char) This function returns true if the character “char” is a lower case letter [a-z]

V. isspace(char) This function returns true if character “char” is a space, tab, new line, vertical tab, or a carriage return. Basically anything that represents spacing.