**Computer Science 1106 Programming Assignment 1:**

**Exercise One: Simulation Program**

A circular slab of uniform thickness (5 units) (cross-section of the slab is illustrated below) is to be used to shield a nuclear reactor. A radioactive particle entering the shield follows a random path by moving forward, backward, left, or right with equal likelihood, in jumps of one unit (You may assume the radioactive particle begins in the shield( position one) ).

A change in the direction of the particle is interpreted as a collision with a Lead atom in the shield. After 10 collisions the particle’s energy has dissipated and will not escape the shield.

If the particle moves forward 6 positions (without exceeding 10 collisions), it has escaped the shielding. If the particle returns to the reactor, it is has not escaped the shield.

Write a program to simulate 1000 particles entering the shield and determine what percentage of the particles escape the shielding. **Note: The simulation for each particle will end when any of the events described above has occurred.**

# Escaped

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Wall Thickness |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Inside the reactor

Below is a simulation of a particles movement while in the reactor wall.

# Escaped

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Wall Thickness |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Inside the reactor

Note: The slab encompasses the entire core of the reactor so it is possible that a simulation might never end.

**Output: % of escaped particles**

**Input: N/a**

**Process: for each particle simulate ¼ rng while it is between the lead walls and if it changed directions less than or equal to 10 times**

**For each particle count if it escaped**

**Calculate rate of escaped particles**

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <iomanip>

using namespace std;

const int TOTAL\_PARTICLES = 1000; //number of particles to run the simulation for

const int FORWARD = 1;

const int BACKWARD = 2;

const int DIR\_SENTINEL = -1; //represents the particle has not yet moved

const int OUTER\_BOUND = 5; //represents the outer layer of the lead shield

const int INNER\_BOUND = 1; //represents the inner layer of the lead shield

const int MAX\_COLLISIONS = 10;

int main()

{

int escapedParts = 0,

direction, collisions, prevDirection, partPos;

double rateEscaped;

bool isPartActive, isInShield;

srand(static\_cast<int>(time(0))); //seed rng

for (int particle = 1; particle <= TOTAL\_PARTICLES; particle++) //loop for each particle

{

prevDirection = DIR\_SENTINEL; //reset previous Direction

collisions = MAX\_COLLISIONS; //collission counter, counts to 0

partPos = INNER\_BOUND; //the particle will disapate

isPartActive = true; //boolean test variables start activated

isInShield = true;

while (isPartActive && isInShield)

{

direction = rand() % 4 + 1; //generate a direction 1 to 4

if (direction != prevDirection && prevDirection != DIR\_SENTINEL) //test for change in direction

collisions--;

switch (direction) //move particle forward or backwards

{

case FORWARD:

partPos++;

break;

case BACKWARD:

partPos--;

break;

}//End Switch

prevDirection = direction; //update previous direction

isPartActive = static\_cast<bool>(collisions); //single particle simulation ends when particle disapates due to collisions OR

isInShield = partPos >= INNER\_BOUND && partPos <= OUTER\_BOUND; //when particle escapes or re-enters the reactor. THESE CONTROL THE WHILE LOOP

}//End While

if (partPos > OUTER\_BOUND) //test if the end condition was caused by an escaped particle

escapedParts++;

}//End For

rateEscaped = static\_cast<double>(escapedParts) / TOTAL\_PARTICLES \* 100; //calculate escaped particles percentage

cout << " Total Particles: " << setw(5) //display formatted results

<< TOTAL\_PARTICLES << endl;

cout << " Escaped Particles: " << setw(5)

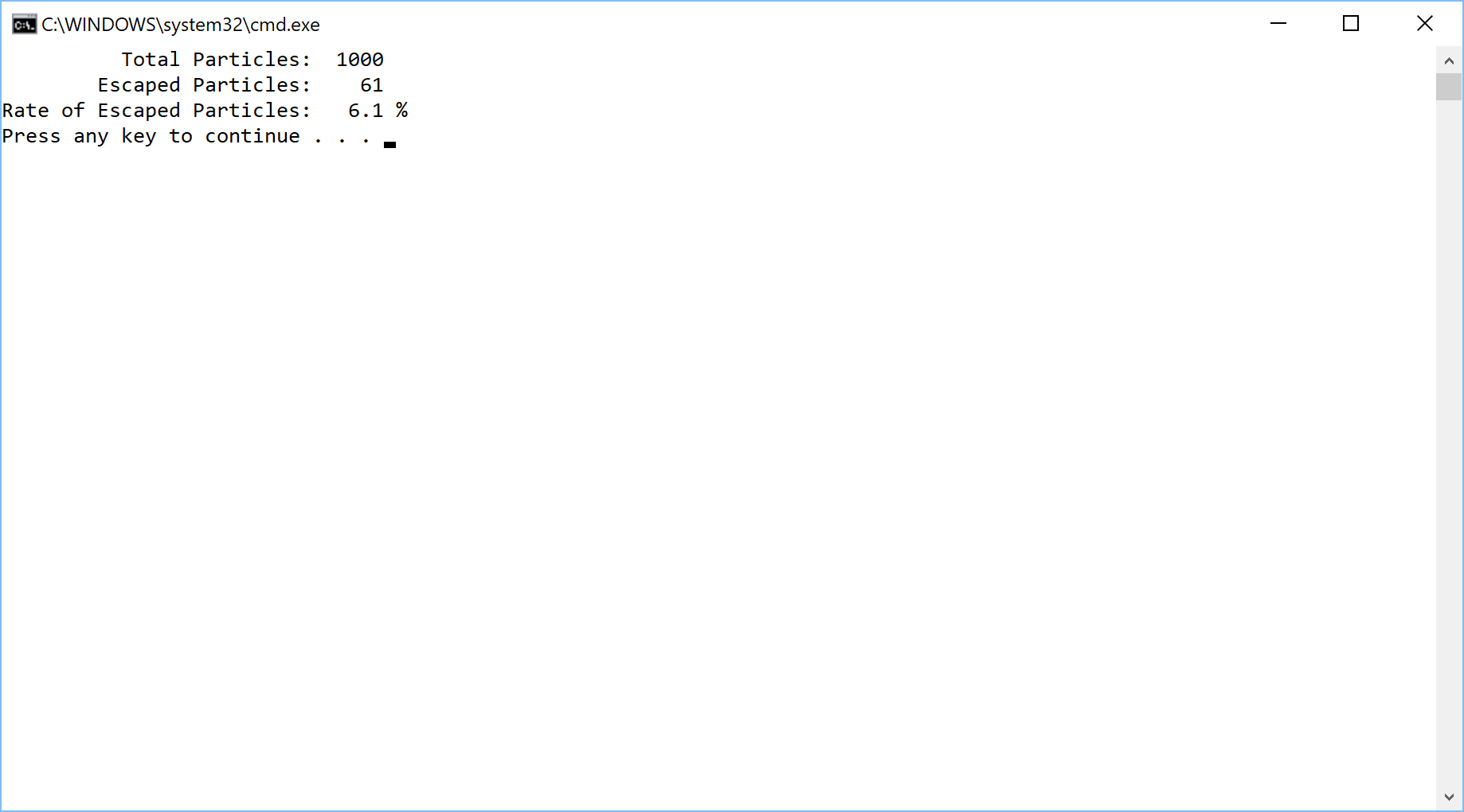
<< escapedParts << endl;

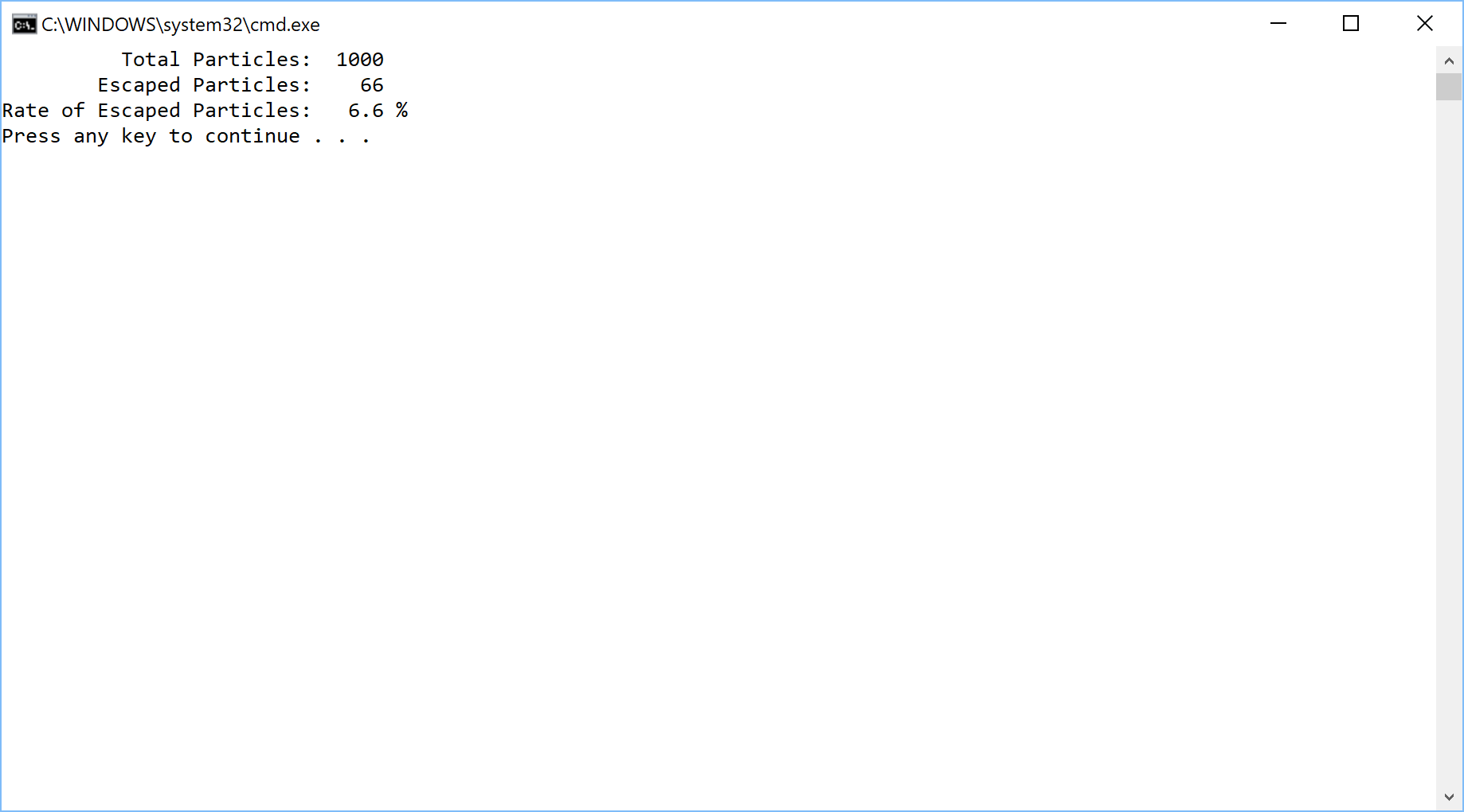
cout << "Rate of Escaped Particles: " << setw(5)

<< rateEscaped << " %" << endl;

return 0;

}





**Exercise 2:**

Write a C++ program that reads a file and counts the number of alphabetic characters, number of uppercase letters, number of punctuation characters, and the number of whitespace characters in the file. You must utilize the get function to process the file one character at a time. The data file to use is the Gettysburg Address, which is found below. Your program should utilize the **cctype** functions found [here](http://www.cplusplus.com/reference/clibrary/cctype/) and on pages 548-553 in your textbook. The author’s explanation is very good. The program should produce a report containing the above information.

Four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in Liberty, and dedicated to the proposition that all men are created equal.

Now we are engaged in a great civil war, testing whether that nation, or any nation so conceived and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to dedicate a portion of that field, as a final resting place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this.

But, in a larger sense, we cannot dedicate -- we cannot consecrate -- we cannot hallow -- this ground. The brave men, living and dead, who struggled here, have consecrated it, far above our poor power to add or detract. The world will little note, nor long remember what we say here, but it can never forget what they did here. It is for us the living, rather, to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us -- that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion -- that we here highly resolve that these dead shall not have died in vain -- that this nation, under God, shall have a new birth of freedom -- and that government of the people, by the people, for the people, shall not perish from the earth.

**Problem Analysis:**

Output:

Number alphabetic characters

Number of uppercase letters

Number of punctuation chars

Number of whitespace chars

Input:

Text File

Process:

Get textfile

while input buffer not empty process file 1 char at a time

process character to and count char type

close textfile

#include <iostream>

#include <fstream>

#include <iomanip>

#include <cctype>

#include <string>

#include <cstdlib>

using namespace std;

const string INPUT\_FILE\_NAME = "GettysburghAddress.txt";

int main()

{

ifstream inputFile;

int numAlpha = 0,

numUpper = 0,

numPunct = 0,

numWhitespace = 0;

char ch;

inputFile.open(INPUT\_FILE\_NAME);

if (!inputFile)

{

cout << "Input file failed." << endl;

exit(1106);

}

while (inputFile.get(ch))

{

if (isalpha(ch))

numAlpha++;

if (isupper(ch))

numUpper++;

if (ispunct(ch))

numPunct++;

if (isspace(ch))

numWhitespace++;

}

cout << "\t File Contents" << endl;

cout << "\t --------------------" << endl;

cout << "\t Alphabetic characters: " << setw(6) << numAlpha << endl;

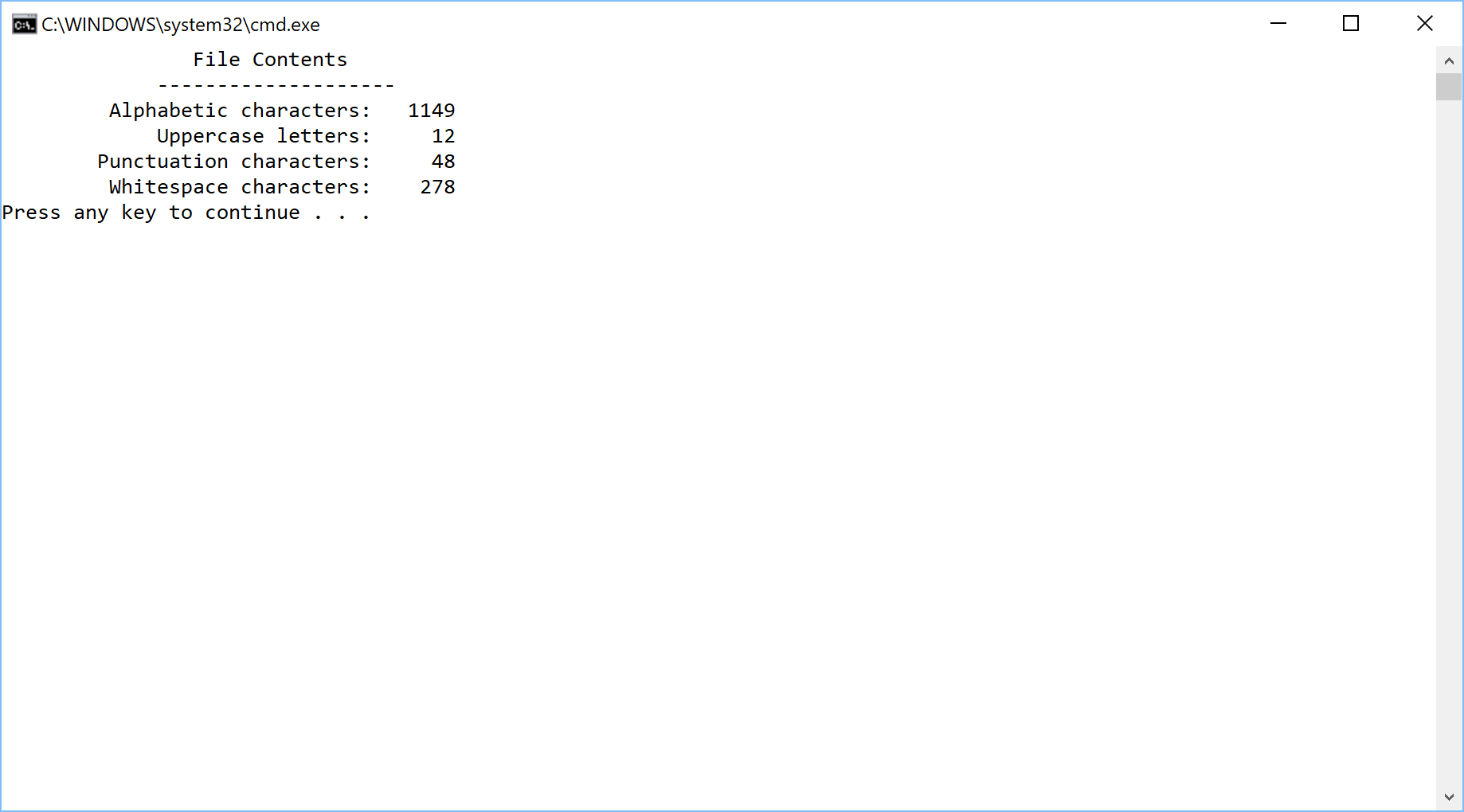
cout << "\t Uppercase letters: " << setw(6) << numUpper << endl;

cout << "\tPunctuation characters: " << setw(6) << numPunct << endl;

cout << "\t Whitespace characters: " << setw(6) << numWhitespace << endl;

return 0;

}



**What to turn in:**

1. For each exercise copy and paste your source code into a word document and paste screen shots of the console window of your program directly under your source code in the document. Also paste a screen shot of the output window showing a successful compilation.
2. A hard copies of the source code for each program.
3. Compress all files into **a single file** with the filename: yournameProgram1.zip
4. Place the compressed file into the D2L drop box called Programming Assignment One.