CMSC 202 Spring 2020

Project 4 – UMBC Cipher

Assignment: Project 4 – UMBC Cipher

Due Date: Tuesday, April 21st at 8:59pm

Value: 80 points

1. Overview

In this project, you will:

- Practice creating classes with inheritance
- Working with pointers as they pertain to classes
- Using polymorphism
- Working with a larger number of classes

2. Background

Protecting data and even more importantly information is a concept that is common throughout headlines today. Articles ripped from today's headlines often read like this "Apple apps ranked as biggest security risk on the PC" or "Are people really your biggest cyber security risk?". So, what can we do as computer scientists to help reduce the risk of our data being compromised?

There are three main concepts that are at the heart of information security (Infosec). They are confidentiality, integrity, and availability (CIA for short). Figure 1 shows these information security goals.

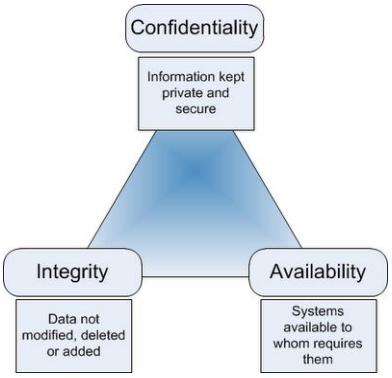

Figure 1. Information Security Goals (Williams, 2012)

Confidentiality: Information kept private and secure.

Integrity: Data not modified, deleted, or added.

Availability: Systems available to whom requires them.

For this project, we are going to focus on confidentiality which is related to keeping our information private and secure. Many current system vulnerabilities are related to not keeping our data confidential such as when Equifax had a data breach or Ashley Madison was hacked.

3. Cryptology, Encryption, and Decryption

As this is just a general introduction to some of the most basic ways to help secure our data, we are not going to delve into the complex ways that we currently employ in the information assurance sector to protect our data. Rather, we are going to introduce some of the key concepts and practice coding a simple encryption/decryption system to protect our data.

Cryptology: The practice and study of protecting our data from unwanted users is what we consider to be cryptology. We have been employing simple cryptologic methods for securing our information for thousands of years. For our project, the first encryption method that we will implement was documented to have been used by Julius Caesar back prior to 44B.C.

Encryption: This is one way to help secure our data by converting it from something readable to something that is apparent nonsense. Converting ***Hello*** to ***Ifmmp*** is an example of a simple 1-character shift (or Caesar Cipher). H->I, e->f, l->m, A->B, and o->p.

Decryption: In order to be able to read something that has been encrypted, we need to reverse the algorithm so that we can read it again. To do this, we decrypt it. In our previous example, we did a 1-character shift from Hello to Ifmmp. To decrypt it, we would need to shift the encrypted message back from Ifmmp to Hello.

As you can see, the idea of *basic* encryption and decryption isn't particularly complicated. Cartoons and cereal boxes have been using things like secret decoder rings as prizes or toys for children.

As we have discussed in class, computers are not particularly interested in storing characters or strings in memory (mostly because they can only speak

binary!) so they convert the characters into a number. This is particularly interesting because they use a *character-encryption* scheme to do this.

4. ASCII

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Table 1. ASCII Table (Wikipedia, 2015)

Now, the notion of character-encryption sounds more complicated than it really is. What we are doing is very simple. We are mapping each character on a standard keyboard to a specific number. If you look at Table 1 above, you will see that the first 32 characters are control characters and therefore do not print anything. The next 95 characters are printable and are the ones that we are most familiar with.

5. Assignment Description

Your assignment is to implement a system that can be used to encrypt or decrypt three different ciphers.

Cipher 1: Caesar Cipher - The first cipher is described above. It is a simple substitution cipher. It is a type of substitution cipher in which each letter in the plaintext (we will read it in as a file) is replaced by a letter some fixed number of positions down the alphabet. For example, if we shifted 3, ‘a’ would be a ‘d’. Our cipher must allow for the user to shift any legal integer number of

places (always to the right). So, if we shifted an 'e' 100 places to the right, it would be 100 % 26.

Cipher 2 – Vigenère Cipher – The second cipher is a version of the Caesar cipher. It is a method of encrypting alphabetic text by using a series of interwoven Caesar ciphers based on the letters of a keyword. Rather than using an integer to indicate the shift, we will use a single word. For additional details: https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

Cipher 3: Ong Cipher - The last cipher is different than the first two. It works kind of like pig latin (and really isn't a cipher). For the ong cipher, for each character in the cipher, you are going to evaluate if it is a vowel or not. If it is a vowel, you will just ignore it. If it is a consonant, you will add "ong" after every consonant.

For example:

UMBC = U-Mong-Bong-Cong

True Grit = Tong-rong-u-e Gong-rong-i-tong

You should include the – between each letter in a word. If you are still confused, look at this presentation: https://prezi.com/nbp4az6wzrmj/how-to-speak-ong/

6. Requirements:

This is a list of the requirements of this application. For this project, it is up to you exactly how you want to implement it. For you to earn all the points, however, you will need to meet all the defined requirements.

- You must follow the coding standard as defined in the CMSC 202 coding standard (found on Blackboard under course materials). This includes comments as required.
- The project must be turned in on time by the deadline listed above.
- The project must be completed in C++. You may not use any libraries or data structures that we have not learned in class. Libraries we have learned include **`<iostream>`**, **`<fstream>`**, **`<iomanip>`**, **`<cmath>`**, **`<cstdlib>`**, **`<vector>`**, **`<ctime>`**, and **`<string>`**. You should only use **`namespace std`**.
- You may use any of the conversion functions such as **`stoi`** or **`atoi`** although you can do this project without either.

- You will want to use stringstream which is a special library **`#include <sstream>`**. It allows you to store a stream as a variable (like if you wanted to STORE a **`cout`** statement). You can use something like:

```
stringstream ss;
ss << myVariable << 'o' << 31 << endl;
return ss.str();
```

- Using the provided files, **`Cipher.h, Caesar.h, Vigenere.h, Ong.h, CipherTool.h`** and **`proj4.cpp`**, create a cipher tool. You can copy the files from my directory in **`/afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj4`**. To copy them, navigate to your project 4 folder and type:

```
cp /afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj4/* .
```

- You may **NOT** modify the headers files.
- There are five required classes in this project: Cipher, Caesar, Vigenere, Ong, and CipherTool. All functions listed in their class files must be implemented completely (even if you do not use them).
- All ciphers must be dynamically allocated (and deallocated).
- All user input must be validated. For example, if a menu allows for 1, 2, or 3 to be entered and the user enters a 4, it will re-prompt the user. However, the user is expected to always enter the correct data type. i.e. If the user is asked to enter an integer, they will. If they are asked to enter a character, they will. You do not need to worry about checking for correct data types.
- The name of the input file may change. We have four sample input files prepared for you to test your program. The name of the files are: **`proj4_test1.txt, proj4_test2.txt, proj4_test3.txt and proj4_test4.txt.`**

  The last input file (**`proj4_test4.txt`**) has one of each of the three ciphers both encrypted and decrypted. An input file can have any number of inputs.
- You will be using a vector to store the cipher pointers. Each cipher (and subtypes) must be dynamically allocated (and deallocated!).

6.1. Class Requirements

- Cipher – Use **`Cipher.h`** to write the **`Cipher.cpp`** file for this class. This class is the parent class for each of the ciphers. It holds a message (the piece of data that will be encrypted and decrypted) and a Boolean indicating if the message is currently encrypted or decrypted.
- Caesar – Use **`Caesar.h`** to write the **`Caesar.cpp`** file for this class. Caesar objects are a specific cipher that uses the Caesar cipher to encrypt and decrypt. It has a variable specific to Caesar objects named **`m_shift`** that is how many characters the cipher shifts the characters.
- Vigenère – Use **`Vigenere.h`** to write the **`Vigenere.cpp`** file for this class. Vigenere objects are a specific cipher that uses the Vigenere cipher to encrypt and decrypt. It has a variable specific to Vigenere named **`m_key`** that is used to encrypt and decrypt.
- Ong – Use **`Ong.h`** to write the **`Ong.cpp`** file for this class. Ong objects are a specific cipher that uses the Ong cipher to encrypt and decrypt. Ong does not require a shift or a key so this class does not have any specialty variables.
- CipherTool – Use **`CipherTool.h`** to write the **`CipherTool.cpp`** file for this class. This is the longest and most complicated of the classes as it manages the entire process. The CipherTool class allows the user to display all ciphers, encrypt all ciphers, decrypt all ciphers, or export all ciphers. The ciphers can be of different types and they are all stored together in a single vector named **`m_ciphers`**.
- Exporting using CipherTool – This project allows users to export ciphers to a new file. This new file should work exactly like one of the provided input files. If you read in a file and then encrypt it, export it and then exit. We should be able to start the project with the exported file and it should work normally. Writing out to a file is very similar to writing out to the console except we have to open and close the file. Examples can be found here: http://www.cplusplus.com/doc/tutorial/files/ Note: The function named: **`FormatOutput`** in each of the child classes formats a stringStream and returns it as a string. We must do this in the subclass because each subclass has special information needed to export (keys or shifts for example).

7. Sample Input and Output

For this project, input files have several parts. The first part indicates the cipher type and will be c for Caesar, v for Vigenere, or o for Ong. The delimiter should be a constant and will be a |. The second piece of information is if the message is encrypted or not. It will be a 1 if encrypted and 0 if not encrypted. Third is the message itself. This is what will be encrypted and decrypted as the project progresses. Messages can have any characters (valid in standard ASCII) except for |. Finally, if the cipher is a Caesar, it will have the shift. If the cipher is a Vigenere it will have the key. If the cipher is an Ong it will have nothing.

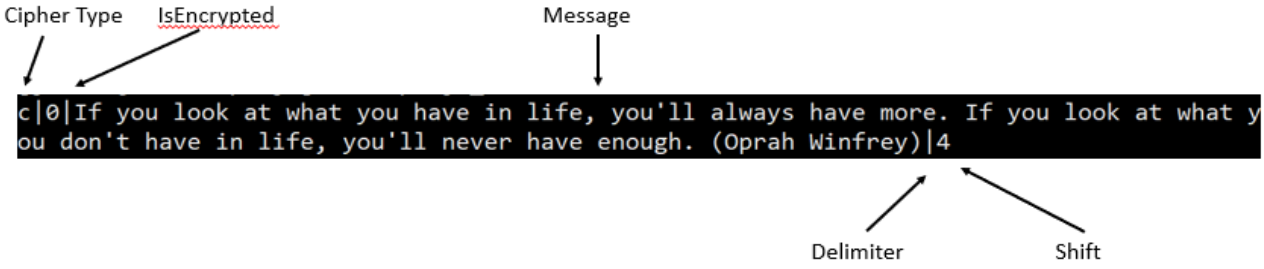

Figure 2. Input Files

For this project, the data files can change based on the ciphers you are going to load. Similar to project 3, included in the main function is the command line arguments to use each of these files.

The first three test files are used to test each of the individual ciphers. **`proj4_test1.txt`** is used to test Caesar ciphers. **`proj4_test2.txt`** is used to test Vigenere ciphers. **`proj4_test3.txt`** is used to test Ong ciphers. The last test file tests both encrypted and decrypted inputs of each type.

For example, below is part of **`proj4_test4.txt`**. As you can see, it has both encrypted and decrypted inputs of different types.

```
c|1|Ktw jajwd rnszyj dtz fwj fslwd dtz qtxj xncyd xjhtsix tk
mfuunsjxx. (Wfqum Bfqit Jrjwxts)|5

o|0|We cannot always build the future of our youth, but we can build
our youth for the future. (Franklin D. Roosevelt)|

v|0|Never leave that till tomorrow which you can do today. (Benjamin
Franklin)|happy
```

The file can be downloaded from Prof. Dixon's data folder by navigating to your project 4 folder and typing the following command:

```
cp /afs/umbc.edu/users/j/d/jdixon/pub/cs202/proj4/* .
```

When you load the project, you will be loading the file at the same time. For example, if I wanted to load the **proj4_test1.txt** file, I could use **./proj4 proj4_test1.txt**. Additionally, the **makefile** has all four already coded. You can use **make run1, make run2 etc.** for the normal runs and **make val1, make val2 etc.** for the valgrind runs.

There are four sample runs included named **proj4_sample1.txt**, **proj4_sample2.txt**, **proj4_sample3.txt**, and **proj4_sample4.txt** each from their respective input files (**proj4_test1.txt = proj4_sample1.txt**).

In the sample output below, user input is colored blue for clarity. After compiling and running proj4, the output would look like this:

```
./proj4 proj4_test1.txt
Welcome to UMBC Encryption
What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
1
1. "If you want to lift yourself up, lift up someone
else. (Booker T. Washington)"(Caesar)


2. "If you tell the truth, you don't have to remember
anything. (Mark Twain)"(Caesar)


What would you like to do?
1. Display All Ciphers
```

```
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
2
2 ciphers Encrypted
What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
1
1. "Li brx zdqw wr oliw brxuvhoi xs, oliw xs vrphrqh
hovh. (Errnhu W. Zdvklqjwrq)"(Caesar)

2. "Nk dtz yjqq ymj ywzym, dtz its'y mfaj yt wjrjrgjw
fsdymnsl. (Rfwp Ybfns)"(Caesar)

What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
3
2 ciphers Decrypted
What would you like to do?
1. Display All Ciphers
```

```
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
1
1. "If you want to lift yourself up, lift up someone
else. (Booker T. Washington)"(Caesar)


2. "If you tell the truth, you don't have to remember
anything. (Mark Twain)"(Caesar)


What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
4
What would you like to call the export file?
proj4_test1_output.txt
2 ciphers exported
What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
5
Thanks for using UMBC Encryption
```

```
[jdixon@linux4 proj4]$
```

Here is another run from proj4_test4.txt including the valgrind results.

```
valgrind ./proj4 proj4_test4.txt
==5979== Memcheck, a memory error detector
==5979== Copyright (C) 2002-2017, and GNU GPL'd, by
Julian Seward et al.
==5979== Using Valgrind-3.15.0 and LibVEX; rerun with
-h for copyright info
==5979== Command: ./proj4 proj4_test4.txt
==5979==
Welcome to UMBC Encryption
What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
1
1. If you look at what you have in life, you'll always
have more. If you look at what you don't have in life,
you'll never have enough. (Oprah Winfrey)(Caesar)


2. Ktw jajwd rnszyj dtz fwj fslwd dtz qtxj xncyd
xjhtsix tk mfuunsjxx. (Wfqum Bfqit Jrjwxts)(Caesar)


3. We cannot always build the future of our youth, but
we can build our youth for the future. (Franklin D.
Roosevelt)(Ong)


4. Pong-e-a-cong-e bong-e-gong-i-nong-song wong-i-
```

```
tong-hong a song-mong-i-long-e-. (-Mong-o-tong-hong-e-
rong Tong-e-rong-e-song-a-)(Ong)


5. Never leave that till tomorrow which you can do
today. (Benjamin Franklin)(Vigenere)


6. Lt rdkcv a eueyw dcdl mi cmxryje rr srdnb xp rr
ymxr cqekleq, eur hvcq mmue rr srdnb xp rr ymxr
duicqdq. (M. K. Prwjlne)(Vigenere)


What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
2
3 ciphers Encrypted
What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
1
1. Mj csy psso ex alex csy lezi mr pmji, csy'pp epaecw
lezi qsvi. Mj csy psso ex alex csy hsr'x lezi mr pmji,
csy'pp riziv lezi irsykl. (Stvel Amrjvic)(Caesar)


2. Ktw jajwd rnszyj dtz fwj fslwd dtz qtxj xncyd
xjhtsix tk mfuunsjxx. (Wfqum Bfqit Jrjwxts)(Caesar)
```

```
3. Wong-e cong-a-nong-nong-o-tong a-long-wong-a-yong-
song bong-u-i-long-dong tong-hong-e fong-u-tong-u-
rong-e o-fong o-u-rong yong-o-u-tong-hong-,ong bong-u-
tong wong-e cong-a-nong bong-u-i-long-dong o-u-rong
yong-o-u-tong-hong fong-o-rong tong-hong-e fong-u-
tong-u-rong-e-. (-Fong-rong-a-nong-kong-long-i-nong
Dong-. Rong-o-o-song-e-vong-e-long-tong-)(Ong)


4. Pong-e-a-cong-e bong-e-gong-i-nong-song wong-i-
tong-hong a song-mong-i-long-e-. (-Mong-o-tong-hong-e-
rong Tong-e-rong-e-song-a-)(Ong)


5. Uektp sepkc ahpi rpla imtoggmd wwxao ydj ahn sd
rvdpn. (Zlnypkpn Ugyukaxl)(Vigenere)


6. Lt rdkcv a eueyw dcdl mi cmxryje rr srdnb xp rr
ymxr cqekleq, eur hvcq mmue rr srdnb xp rr ymxr
duicqdq. (M. K. Prwjlne)(Vigenere)


What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
3
6 ciphers Decrypted
What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
```

```
4. Export All Ciphers
5. Quit
1
1. If you look at what you have in life, you'll always
have more. If you look at what you don't have in life,
you'll never have enough. (Oprah Winfrey)(Caesar)


2. For every minute you are angry you lose sixty
seconds of happiness. (Ralph Waldo Emerson)(Caesar)


3. We cannot always build the future of our youth, but
we can build our youth for the future. (Franklin D.
Roosevelt)(Ong)


4. Peace begins with a smile. (Mother Teresa)(Ong)


5. Never leave that till tomorrow which you can do
today. (Benjamin Franklin)(Vigenere)


6. It takes a great deal of courage to stand up to
your enemies, but even more to stand up to your
friends. (J. K. Rowling)(Vigenere)


What would you like to do?
1. Display All Ciphers
2. Encrypt All Ciphers
3. Decrypt All Ciphers
4. Export All Ciphers
5. Quit
5
Thanks for using UMBC Encryption
```

```
==5979==
==5979== HEAP SUMMARY:
==5979==     in use at exit: 0 bytes in 0 blocks
==5979==   total heap usage: 339 allocs, 339 frees,
95,079 bytes allocated
==5979==
==5979== All heap blocks were freed -- no leaks are
possible
==5979==
==5979== For lists of detected and suppressed errors,
rerun with: -s
==5979== ERROR SUMMARY: 0 errors from 0 contexts
(suppressed: 0 from 0)
[jdixon@linux5 proj4]$
```

8. Compiling and Running

You need to write a **`makefile`** for this project as one will not be provided for you in order to compile and test. Make sure to create any macros to help!

Once you have compiled using your **`makefile`**, enter the command **`make run`** or **`./proj4`** to run your program. Make sure you have implemented that as part of your make file. If your executable is not proj4, you will lose points. It should look similar to the sample output provided above. Some of the simulation is based on randomness so the output will vary.

Because we are using a significant amount of dynamic memory for this project, you are required to manage any memory leaks that might be created. Remember, in general, for each item that is dynamically created, it should be deleted using a destructor.

One way to test to make sure that you have successfully removed any of the memory leaks is to use the **`valgrind`** command.

Since this project makes extensive use of dynamic memory, it is important that you test your program for memory leaks using **`valgrind`**:

```
valgrind ./proj4 proj4_test4.txt
```

Note: If you accidently use **`valgrind make run1`**, you may end up with some memory that is still reachable. Do not test this – test using the command above where you include the input file. The makefile should include **`make val1`** (which is ok).

If you have no memory leaks, you should see output like the following:

```
==5606==
==5606== HEAP SUMMARY:
==5606==     in use at exit: 0 bytes in 0 blocks
==5606==   total heap usage: 87 allocs, 87 frees, 10,684 bytes
allocated
==5606==
==5606== All heap blocks were freed -- no leaks are possible
==5606==
==5606== For counts of detected and suppressed errors, rerun
with: -v
==5606== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 6
from 6)
```

The important part is “in use at exit: 0 bytes 0 blocks,” which tells me all the dynamic memory was deleted before the program exited. If you see anything other than "0 bytes 0 blocks" there is probably an error in one of your destructors. We will evaluate this as part of the grading for this project.

Additional information on **`valgrind`** can be found here: http://valgrind.org/docs/manual/quick-start.html

Once you have compiled using the provided **`makefile`**, enter the commands (**`make run1`**, **`make run2, make run3, make run4`**, **`make val1`**, **`make val2, make val3, or make val4`**) or **`./proj4 proj4_test1.txt`** to run your program. If your executable is not proj4, you will lose points. It should look like the sample output provided above.

9. Completing your Project

When you have completed your project, you can copy it into the submission folder. You can copy your files into the submission folder as many times as you like (before the due date). We will only grade what is in your submission folder.

For this project, you should submit **all** (except your makefile) files to the **`proj4`** subdirectory. As you should have already set up your symbolic link for this class, you can just copy your files listed above to the submission folder.

a. cd to your project 2 folder. An example might be cd
   `~/202/projects/proj4`

b. **`cp * ~/cs202proj/proj4`**

You can check to make sure that your files were successfully copied over to the submission directory by entering the command

`ls ~/cs202proj/proj4`

You can check that your program compiles and runs in the **`proj4`** directory, but please clean up any **`.o`** and executable files. Again, do not develop your code in this directory and you should not have the only copy of your program here. Uploading of any **`.gch`** files will result in a severe penalty.

For additional information about project submissions, there is a more complete document available in Blackboard under "Course Materials" and "Project Submission."

IMPORTANT: If you want to submit the project late (after the due date), you will need to copy your files to the appropriate late folder. If you can no longer copy the files into the proj4 folder, it is because the due date has passed. You should be able to see your proj4 files but you can no longer edit or copy the files in to your proj4 folder. (They will be read only)

- If it is 0-24 hours late, copy your files to **`~/cs202proj/proj4-late1`**
- If it is 24-48 hours late, copy your files to **`~/cs202proj/proj4-late2`**
- If it is after 48 hours late, it is too late to be submitted.