**UFO Moviez Assignment Documentations**

**Operating System:** SUSE Linux

**G++ version:** g++ (SUSE Linux) 4.8.5

**Command to compile :** g++ -Wall <program name> -o <output\_file>

Command to execute : ./output\_file

**Question 1:** C arrays have size determined at compile time. Implement a C++ class that provides you with objects that behave like arrays of int except that their size is determined at run time. Explain the reasons for your design decisions

**Explanation**: Vector will start with zero size and zero capacity and it will double the capacity every time if no vacate place available (size == capacity) for new element in array. So, it will grow dynamically as required.  when an element is inserted or deleted, with their storage being handled automatically by the container.

Whenever array is full first double capacity of array allocated then all element of old array is copy into new allocated array and array address is updated with new allocated array.

So, array will grow in double the size as needed. This is so that if we are inserting n items at most only O(log n) regrowth’s are performed and at most O(n) space is wasted.

This Vector class also handle exception cases if user try to access invalid index.

It has overloaded [] and other operator to behave like as normal arrays

**Time complexity:** of common operations on vectors is as follows:

**Random access** - constant O(1)

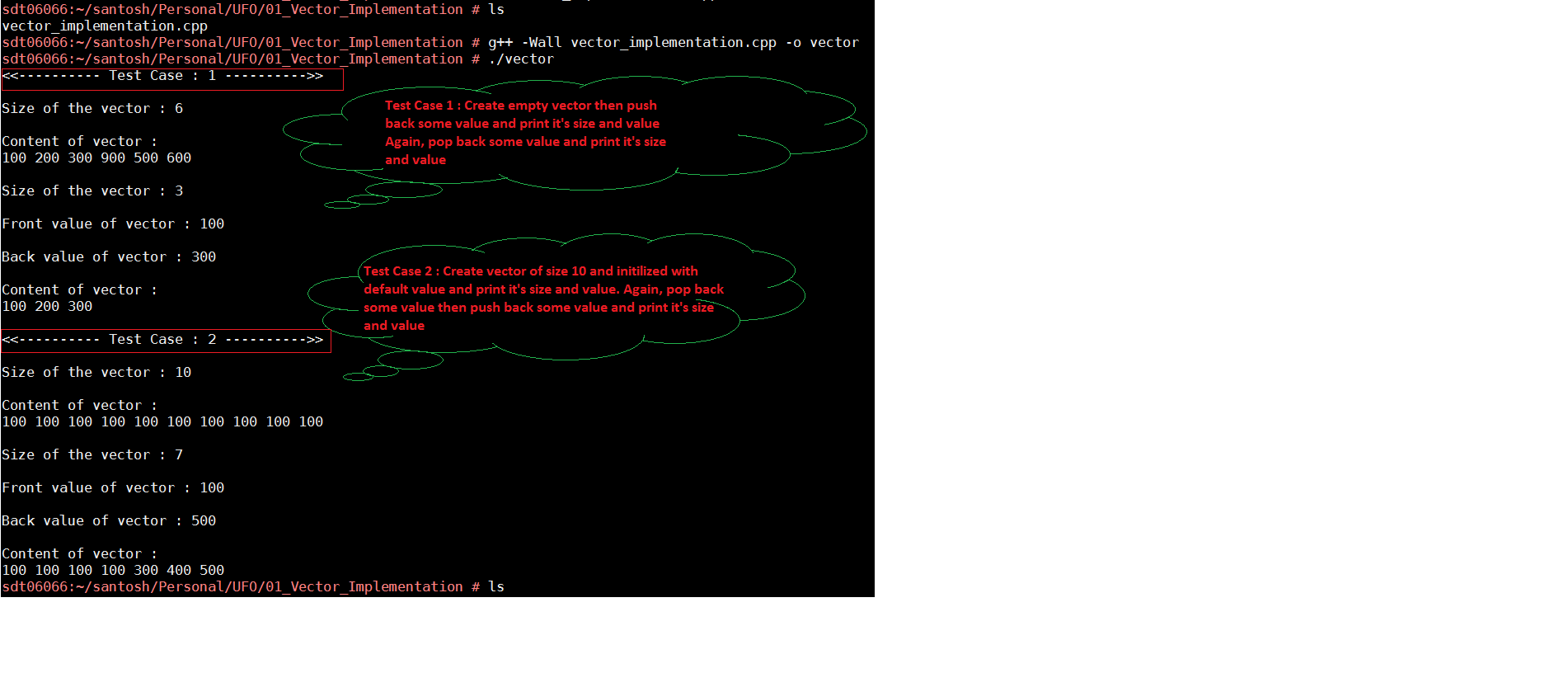
**Insertion or removal** of elements at the end - amortized constant O(1)

**Insertion or removal of elements** - linear in the distance to the end of the vector O(n)

**Compile command:** g++ -Wall vector\_implementation.cpp -o vector

**Execute command:** ./vector

**OUTPUT SCREEN SHOT**



**Question 2:** Design a producer class, a consumer class and a buffer class. Producer reads from a file and writes into a buffer and consumer reads from the buffer and writes into a file. Make the classes thread safe. Using multiple producers merge different files into a single output file. Ask the user for input files. After 3 files, start dumping the combined content into a single put file. Continue to take input files from the user and keep adding them to the process. Ensure that no single producer reading from an infinite source (e.g. /dev/urandom) causes output application to ignore data from the other sources.

**Explanation**:

**Producer class:**

As soon as user enter input file, one producer thread will be created for every input file, that producer thread further creates producer object and call write function to read data from input file and produce to buffer.

PS: Exception will be raised if input file doesn’t exist or don’t have read permission!

**Consumer class:**

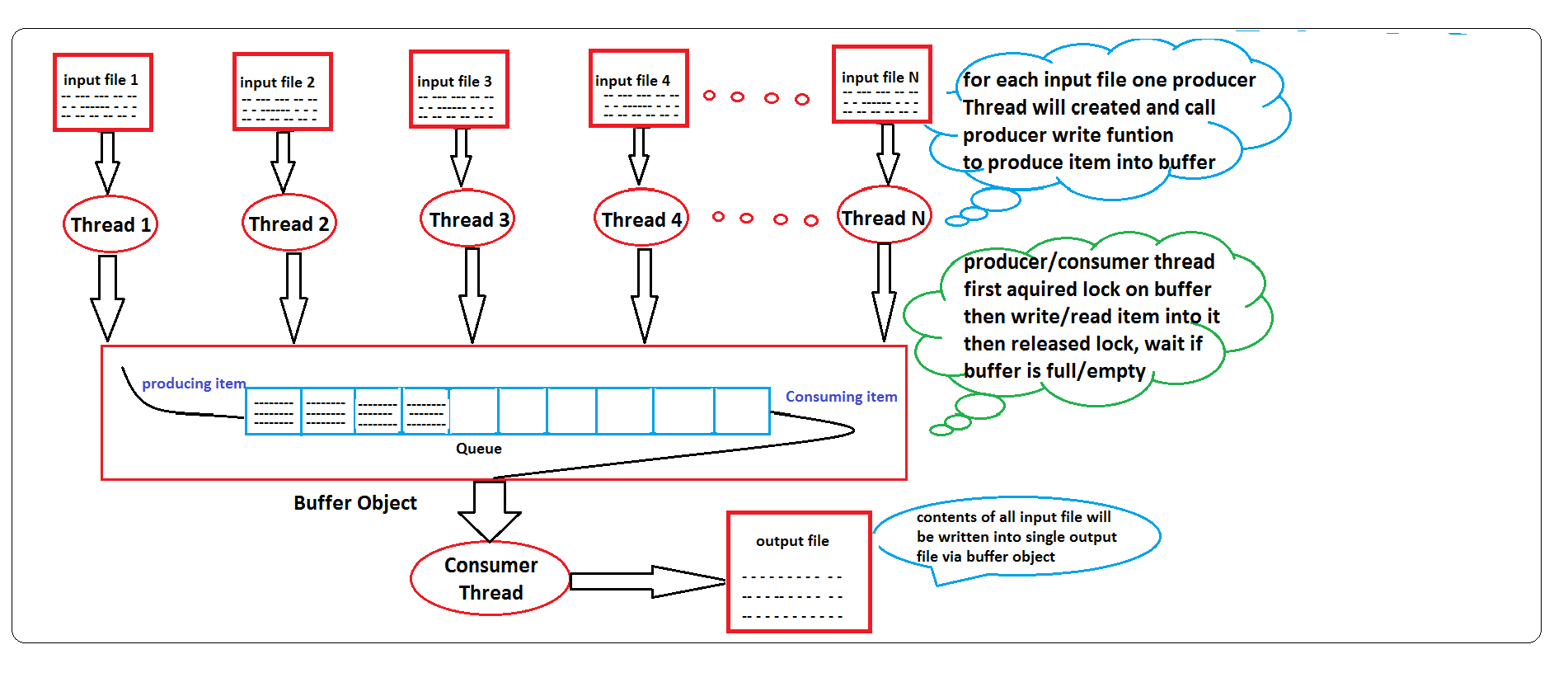
As soon as user enter 3 input files consumer thread will be created that will further creates consumer object and call write function to consume item from buffer and write it into single output file.

PS: Exception will be raised if output directory don’t have write permission.

**Buffer class:**

There will be one buffer object that will be shared by both producer and consumer object. To maintain consistency on buffer in multithreaded environment, each time producer object acquired locked on buffer before write item into buffer and released it after completing writing. Also, Consumer object first acquired locked on buffer than read item and release lock.

**Code Flow Diagram of Producer and consumer:**

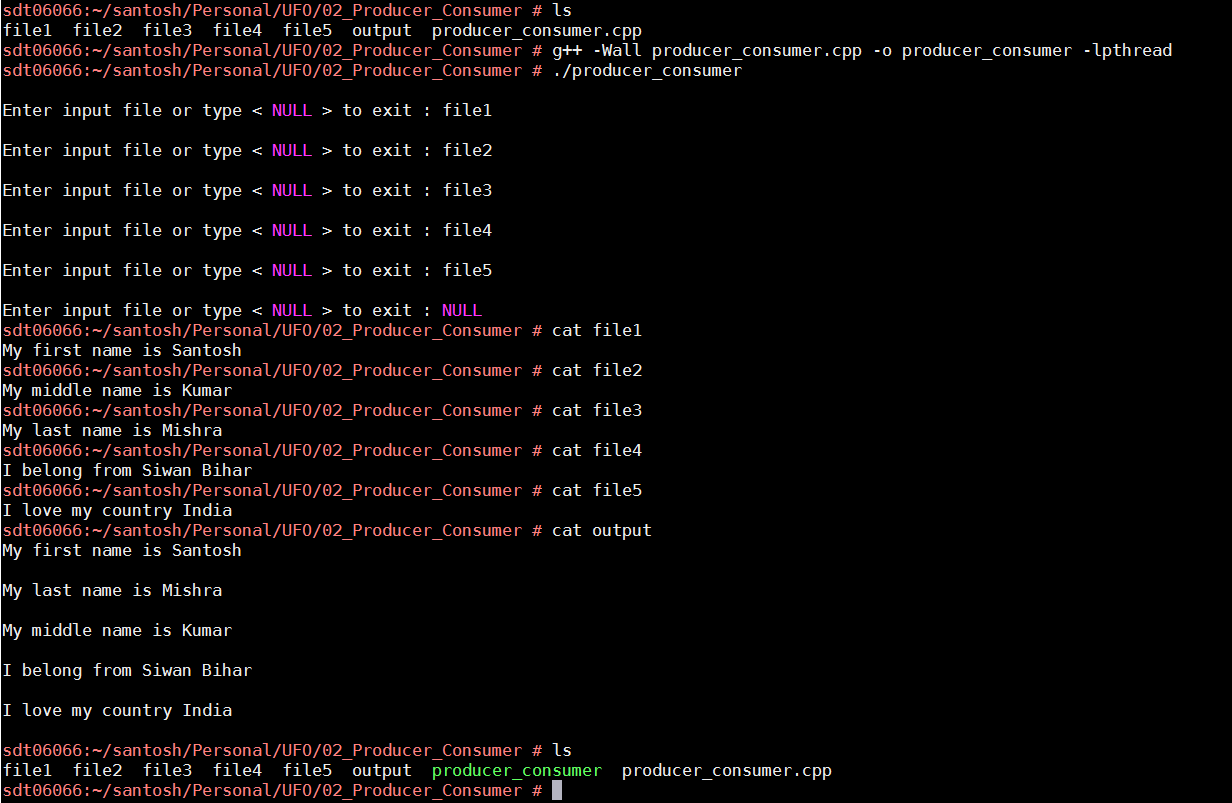


**Compile command:** g++ -Wall producer\_consumer.cpp -o producer\_consumer -lpthread

**Execute command:** ./producer\_consumer

**OUTPUT SCREEN SHOT**

**SCREEN SHOT 1:** Success Cases, it will combine all content of input file into single output file



**SCREEN SHOT 2:** Exception Case, it will be raised exception if input file doesn’t exist! Below screen shot we can see that exception has been generated because user input invalid file name.

