Indian Institute of Technology Bombay, Mumbai
Department of CSE, Kanwal Rekhi Building
CS101 – Computer Programming
Autumn Semester 2014-2015

Lab Handout for Week 6 - 01/09/2014 to 05/09/2014

General Instructions: *Please read the instructions carefully before proceeding further.*
There are programs given in this handout. You need to write and execute them. All the programs that your write and execute need to be uploaded today before leaving the lab. Take help form your TAs to perform the tasks given below:

1. Goto http://www.cse.iitb.ac.in/~cs101
2. Click on "**Lab Assignment Submission**" link.
3. Write your roll number in the text box.
4. Click the "**Choose File**" button.
5. Browse through your directory by navigating to the folder in which you have created the project. Select the program i.e. ".cpp" file, from your project directory.
6. Click the "**Submit**" button.
7. A new page will open with the message "**Upload Successful. Click here to go back**"
8. Perform these steps (1 to 7) for all the programs that you have written.

Objective: In this lab, you are required to solve the practice problems discussed last week during the lectures.

Program 1:
Sieve of Eratosthenes

The Sieve of Eratosthenes is an ancient algorithm to find the prime numbers starting from 2 till a given number (including both). A number is a prime number if it is divisible only by 1 and itself.
In this program, you shall write code that implements the Sieve of Eratosthenes algorithm. The algorithm is described below:

1) We start with a list of consecutive integers from *2* to *n* {2,3,4, … , n}. Assume all elements are initially unmarked.
2) Mark the lowest unmarked element, say *p*.
3) Remove all multiples of *p* in the list., except *p* itself.
4) Repeat steps 2 and 3 until the elements are marked.
5) All elements left in the list are prime numbers less than or equal to *n.*

The skeleton of the main program is given below. You are required to fill in the missing code snippets, compile your code and test it by running while providing as many inputs as you can. Be sure to test for the corner cases: when *n* is a prime number, when *n* is a non-prime number and when *n* is 2.

Code Snippet for Program 1: Sieve of Eratosthenes

```
#include<iostream>
using namespace std;
int main()
{
```

```
int list[101] = {0};  /* Initially "Unmark" all the numbers. */
int i,j;

for (i=2;i<=100;i++)
{
        /* Write code here to "mark" the multiples of "Unmarked" elements
                Remember: Unmarked = 0. Marked = 1
        */
}


        /*Print Unmarked indices */


}
```

Program 2: Queue

Queue is an ordered list that supports only two operations, namely Enqueue and Dequeue.
Enqueue is an insertion operation that inserts that supplied element at the end of the queue. Dequeue is a deletion operation that deletes the element at the start of the queue.
The ends of the queue are called front and rear. Insertions happen at the rear end and deletions happen at the front end of the queue.

Example of implementation: Assume queue to be empty initially.

Operation	Queue
Enqueue(4)	4
Enqueue(5)	4 5
Enqueue(6)	4 5 6
Dequeue()	5 6
Enqueue(8)	5 6 8
Enqueue(9)	5 6 8 9
Dequeue()	6 8 9

Code Snippet for Program 2: Queue

```
#include<iostream>
using namespace std;
int A[10];
int rear = -1; //stores the rear index
int front = -1; //stores the front index
void enqueue(int element)
{
      /* If Queue is full print the error and return */
      /*  if space available at last insert the element
       else shift all elements to start of array & insert the
```

```
        new element */


        /* Alternatively, you can implement a circular queue. i.e. if “rear” is at the end of the array, bring
it to the beginning */
}

void dequeue()
{
       /* check if queue is empty
        remove the front most element */
}
void viewq()
{
       /* print the queue from front to rear*/
}
int main()
{
       A[0] = 3;
       rear++;
       A[1] = 2;
       rear++;  /* Some elements are already in queue */

       char input = 'x';
       int elem;
       while(1)
       {
              cout << "\nEnter option: \ne to Enqueue;\nd to dequeue;\nv to view\nx to exit : ";
              cin >> input;
              if (input == 'e')
              {       cout << "Enter element to insert : ";
                     cin >> elem;
                     enqueue(elem);
              }
              else if(input == 'd') {
                 dequeue(); }
              else if (input == 'x'){  /* Enter 'x' to exit */
                break; }

              else if (input == 'v'){
                viewq(); }
              else {
                cout << "Invalid option"<<endl;
              }
       }
       return 0;
}
```