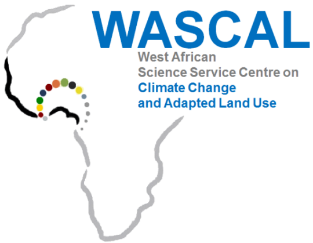
­­­­ 

**CCI-05: Algorithmic and programming**

Project Report:

Student:

**M. Simon Pierre KITEGI**

Plan

[**Description**: 2](#_Toc36931347)

[Modules of the program: 3](#_Toc36931348)

[\*The sum of the first n numbers 3](#_Toc36931349)

[\*The ListType 3](#_Toc36931350)

[\*The StackType 4](#_Toc36931351)

[\*The QueueType 5](#_Toc36931352)

[\*Sorting algorithm 5](#_Toc36931353)

[\*Merging sort 6](#_Toc36931354)

[\*Sequential search 6](#_Toc36931355)

[\*Binary search 6](#_Toc36931356)

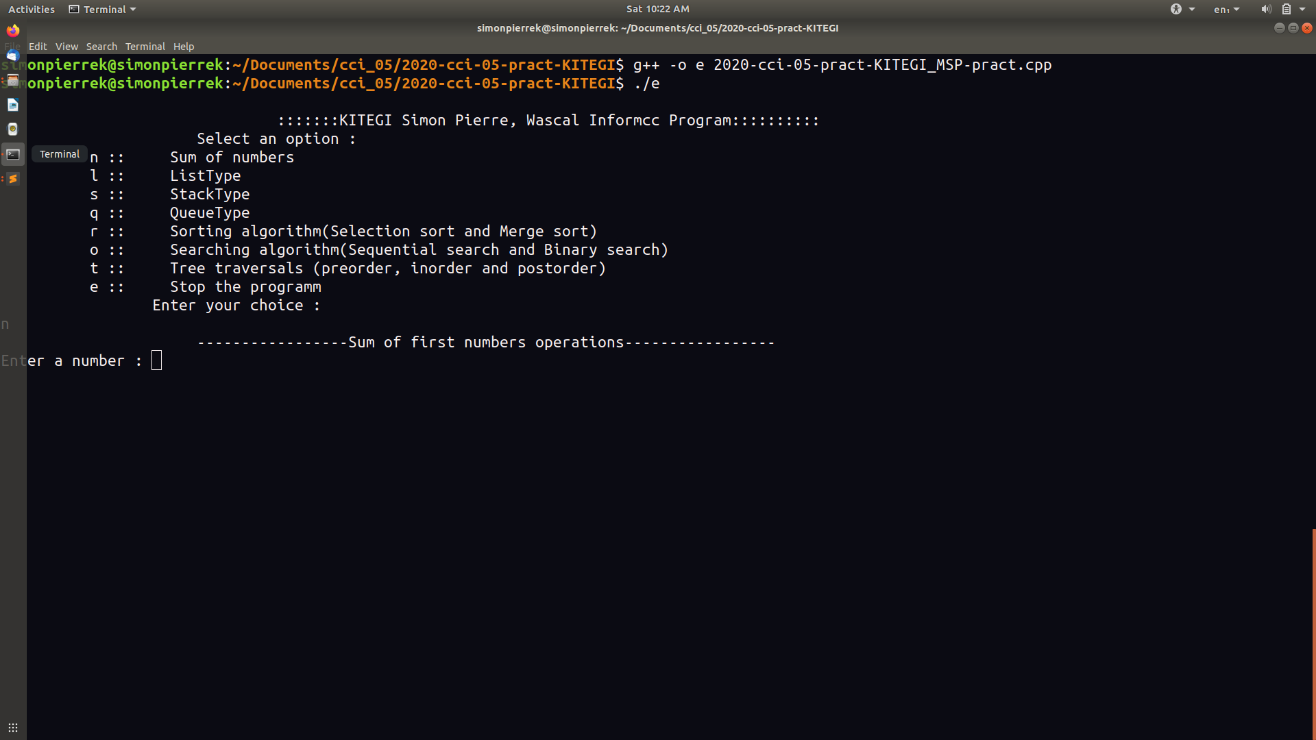
[\*Tree traversals 7](#_Toc36931357)

# **Description:**

The project is a program built to implement abstract data structure and algorithm to search and to sort. These data type are list, stack and circular queue and tree traversal operations. In the next lines we will present each part of the program. The program has been developed under ubuntu in C++.

# **Modules of the program:**

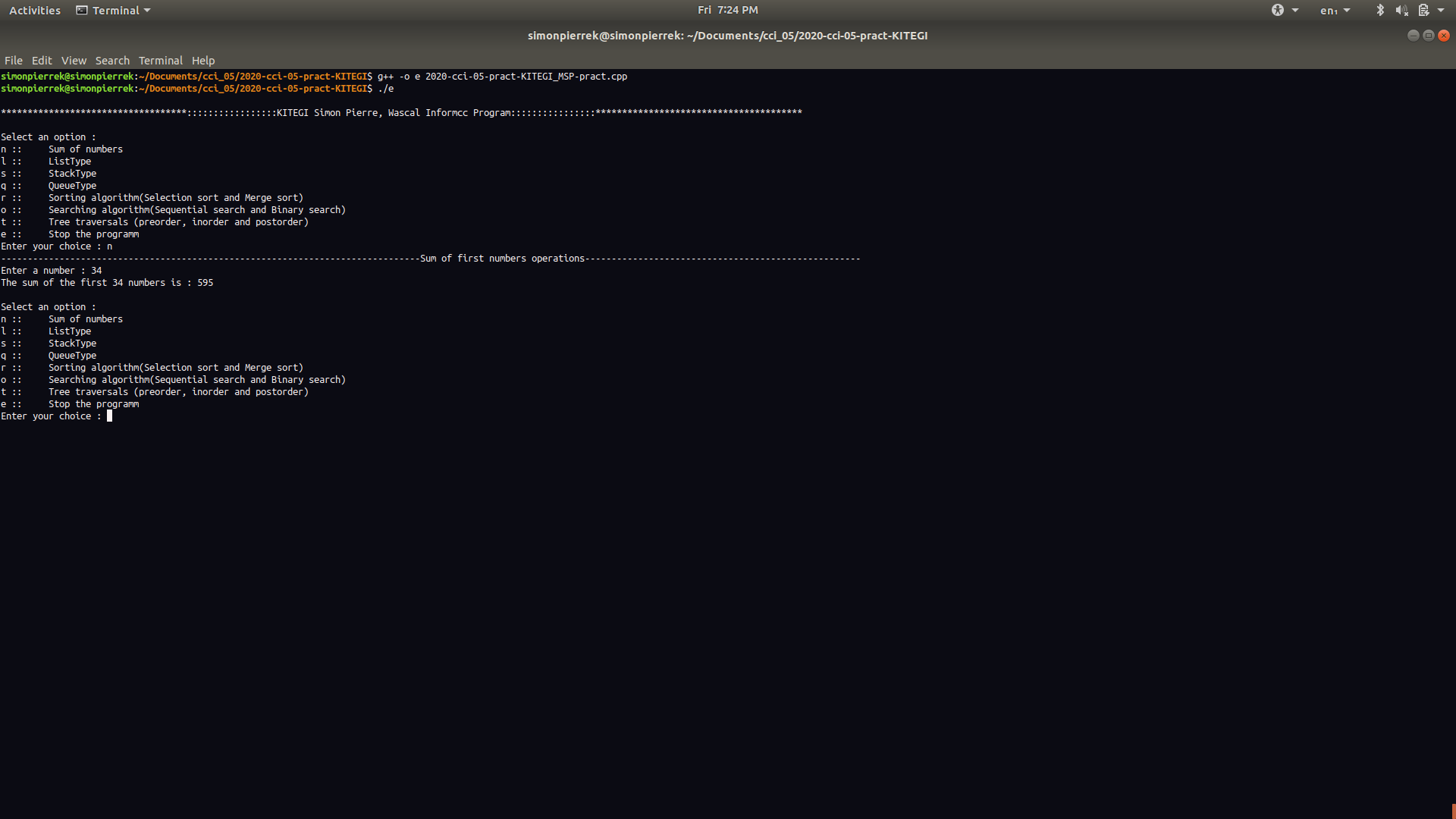
1. The sum of the first n numbers
2. List
3. Stack
4. Queue
5. Sorting algorithm
6. Searching algorithm
7. Tree traversals operations



## \*The sum of the first n numbers

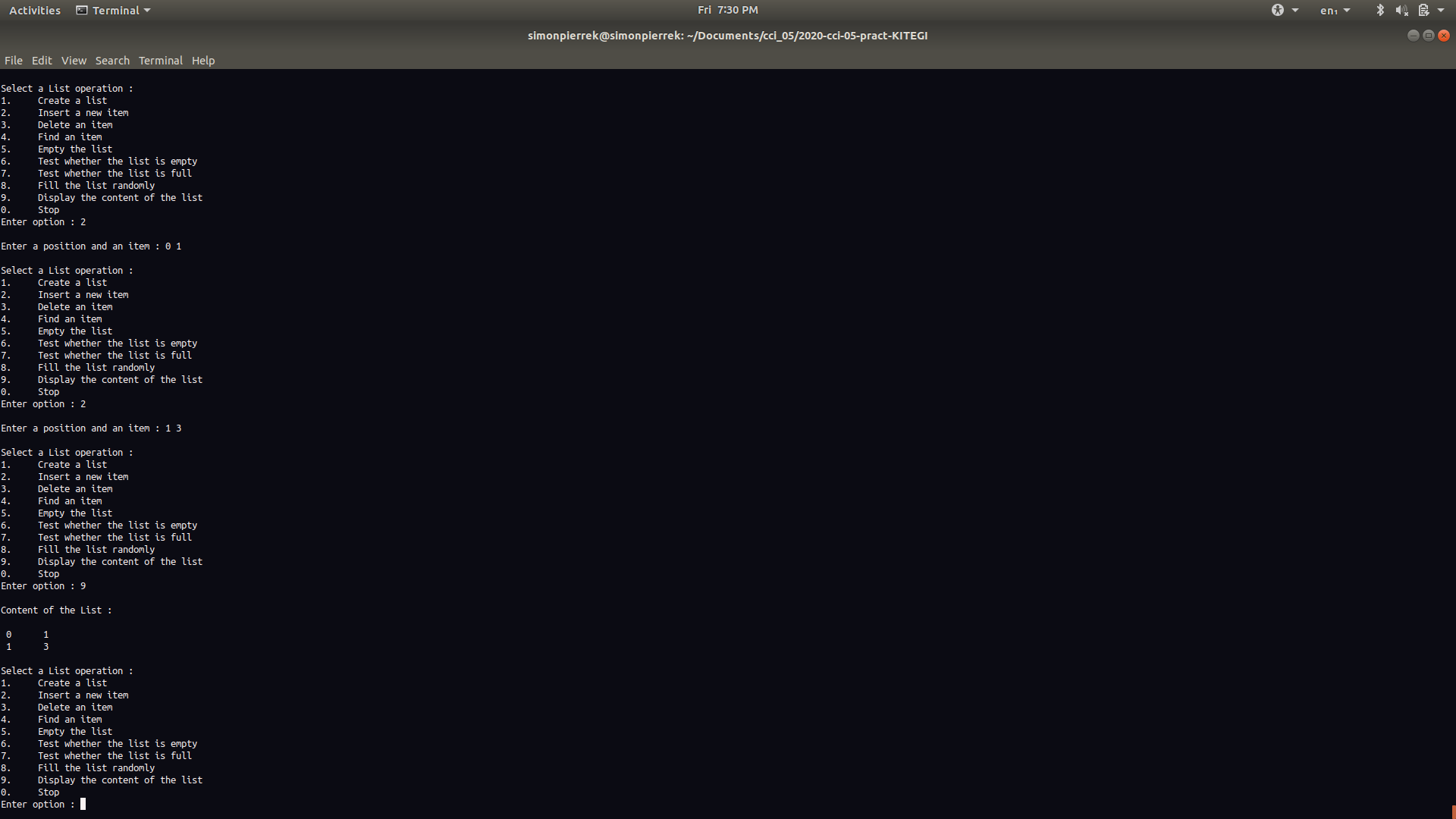
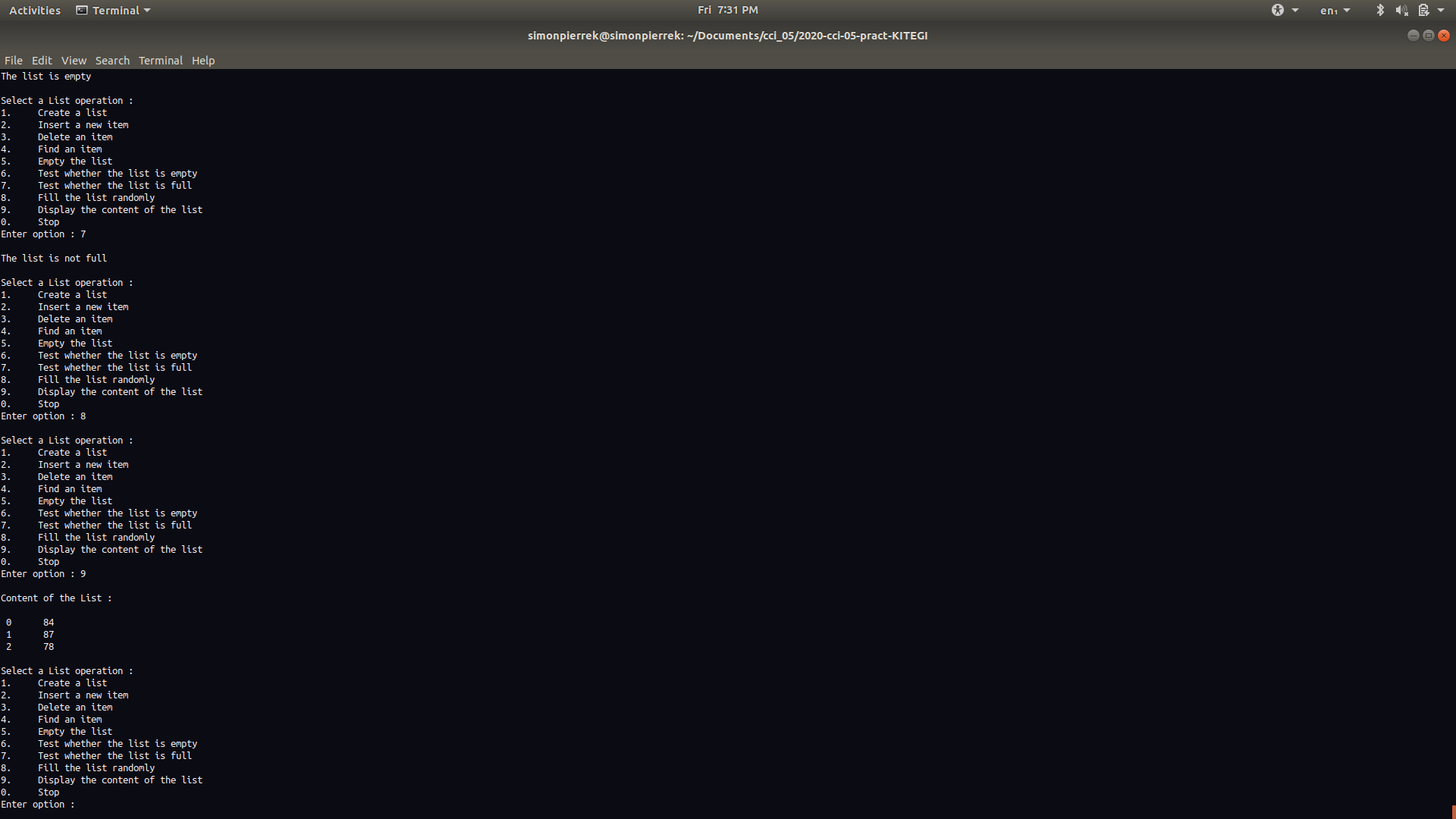
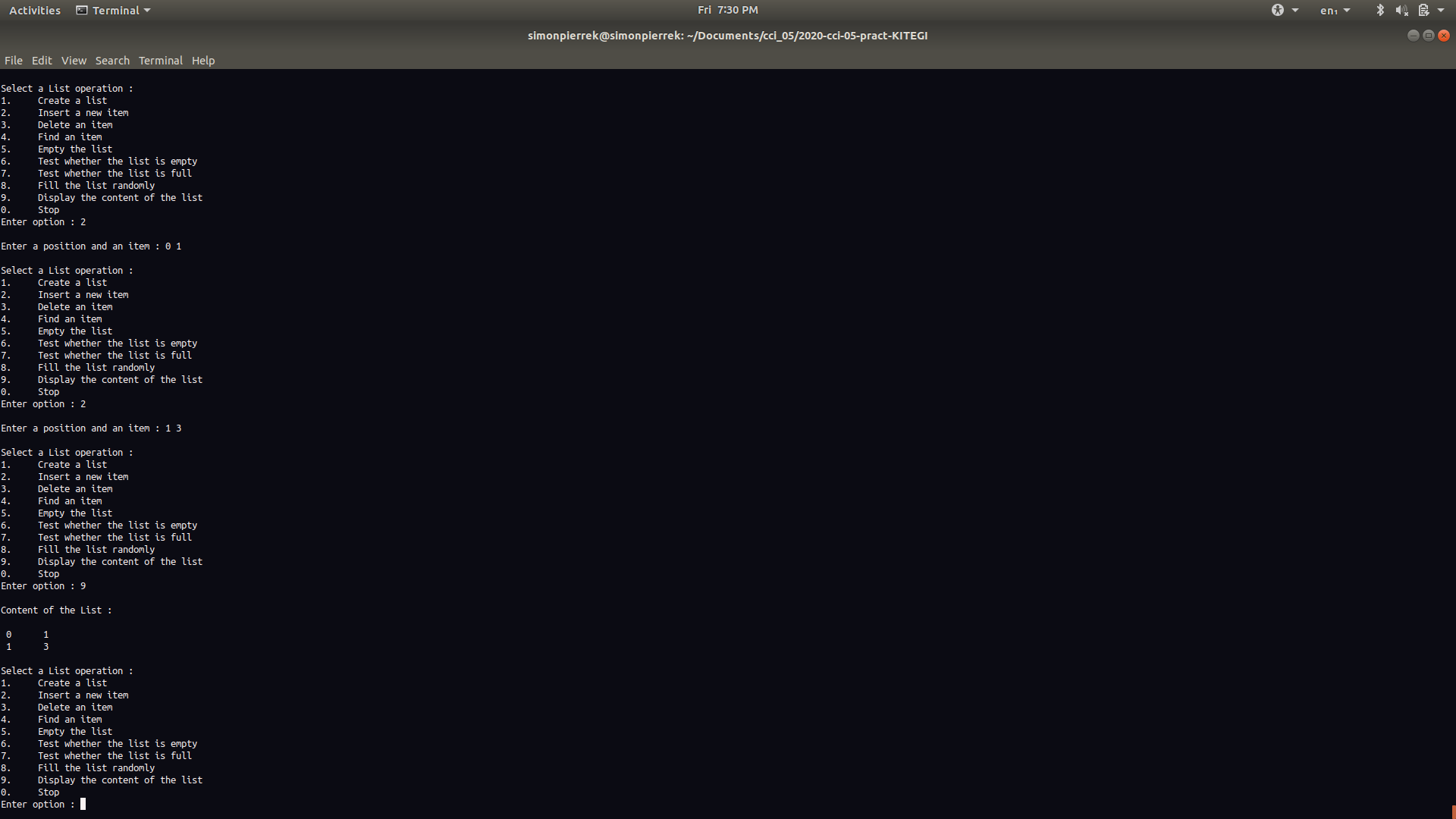
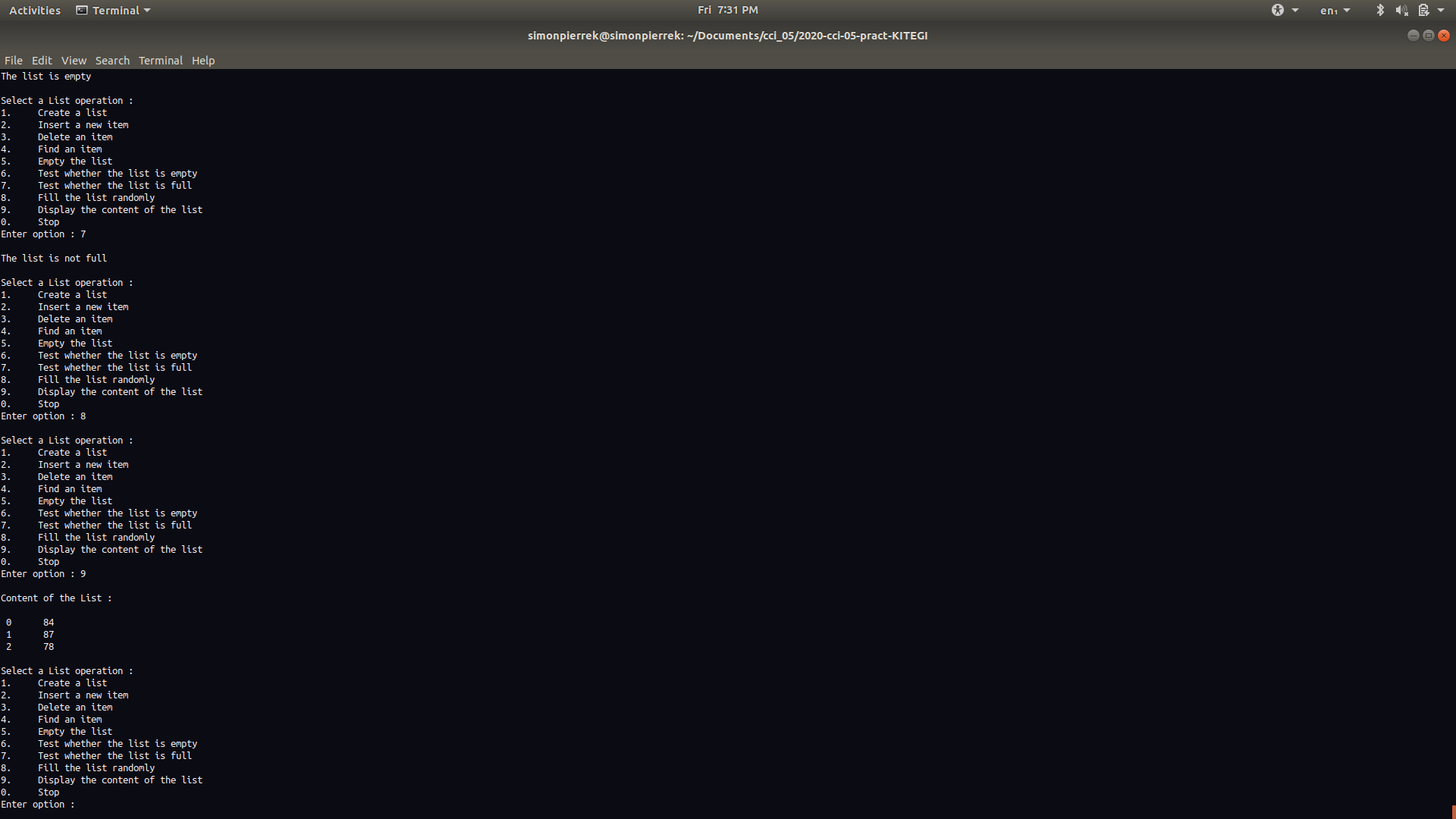
The program will ask the user to enter a given number. Then after writing it, he can press enter to see the result.

In this example we choose 34. And the result is 595.



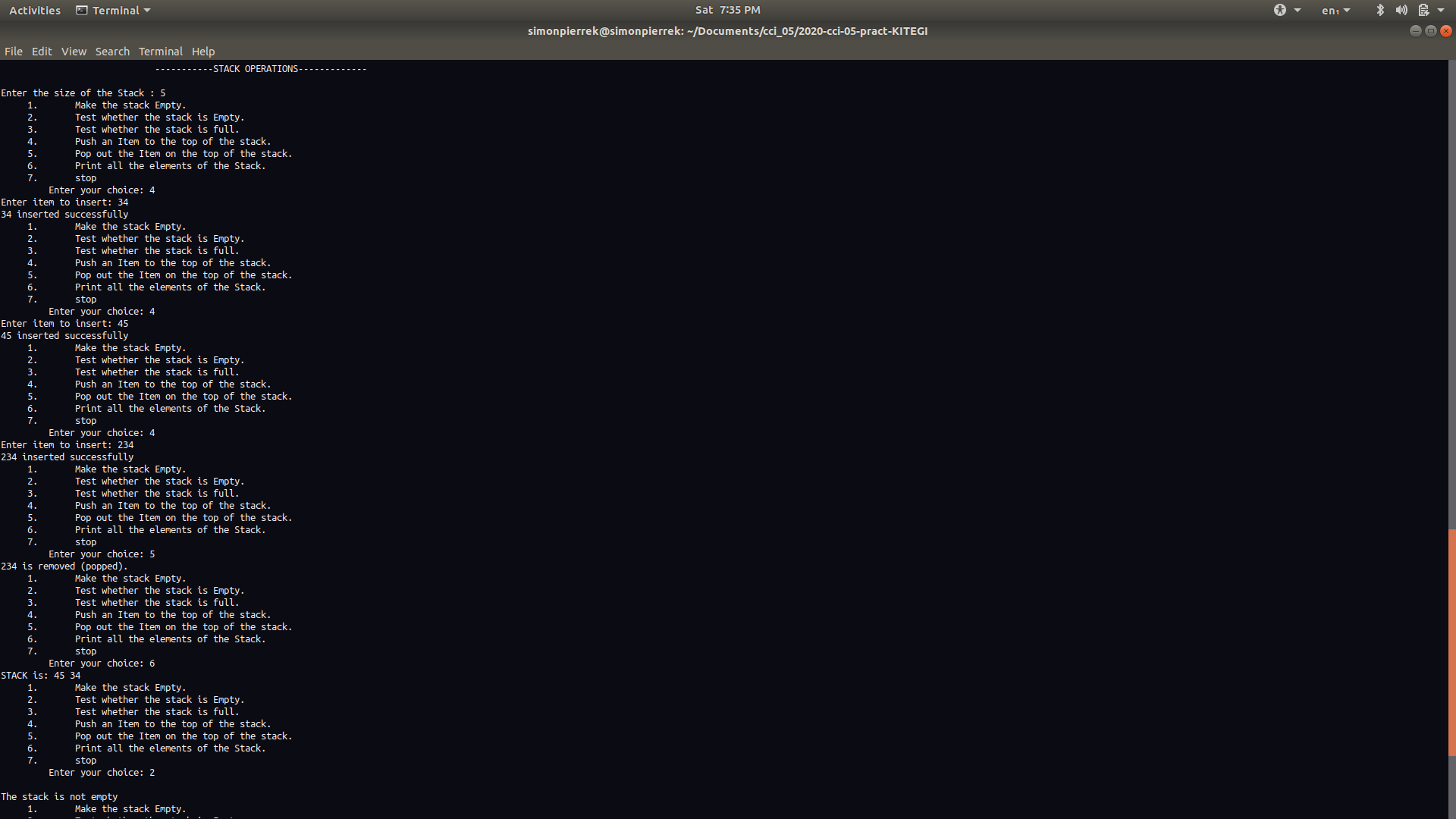
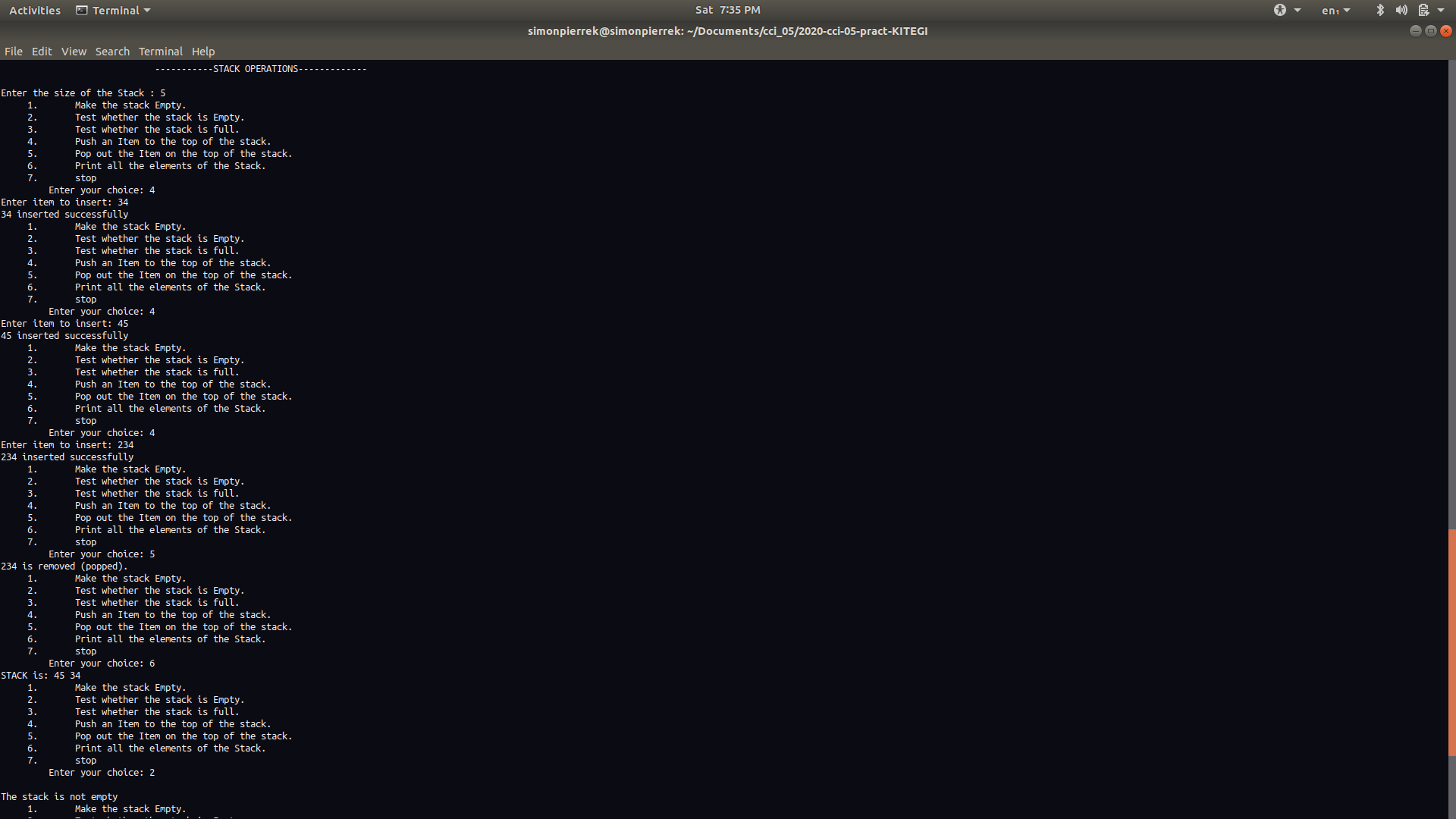
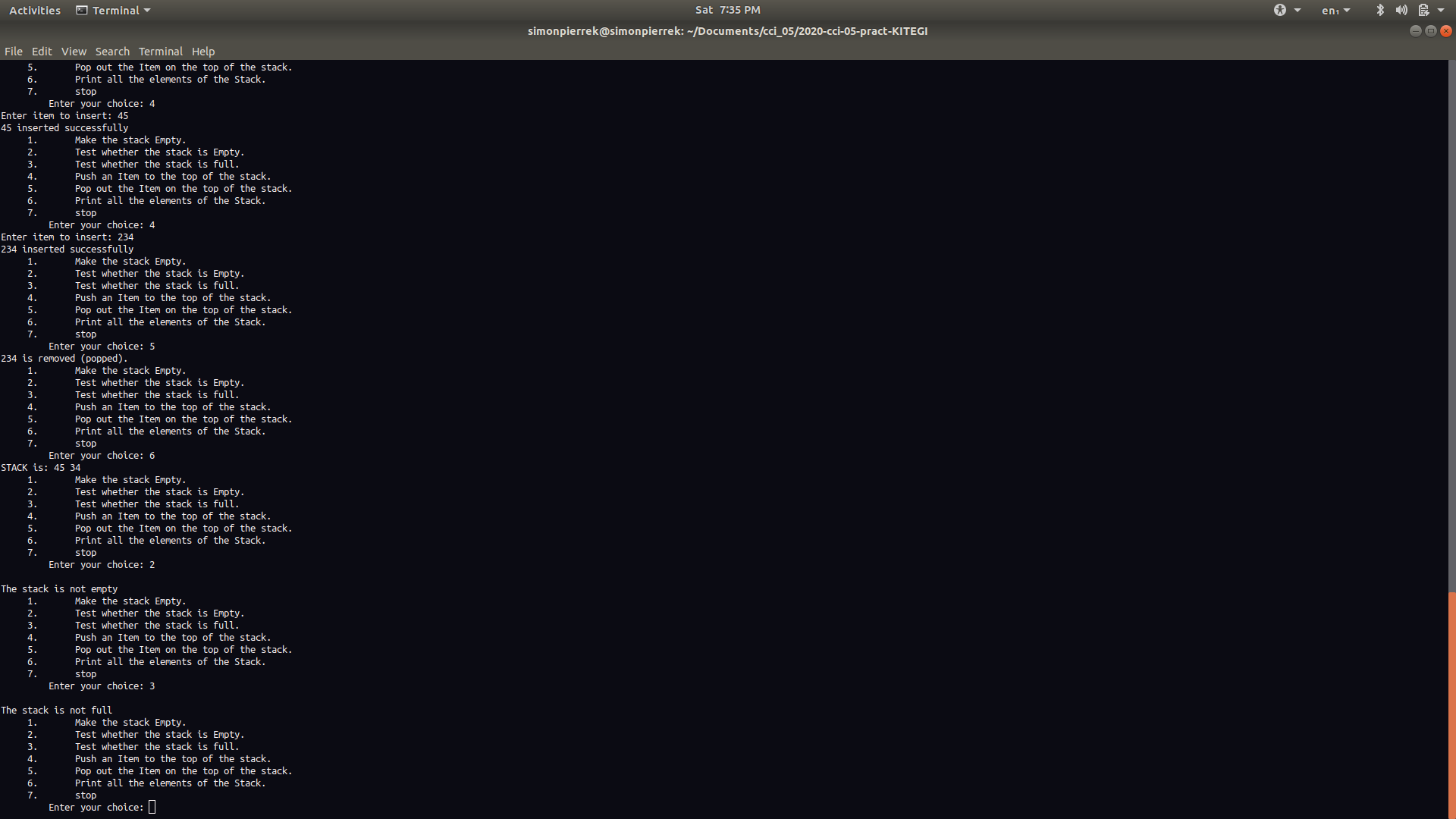
## \*The ListType

The list has a menu of 5 operations structured as this:

Before starting the operation, we have to specify the size of our list. As example, we select 3 as size. Then, we insert 1 and 3 at the positions 0 and 1 respectively. We can display the list with the option 9. With the option 3 we will delete an item by telling the computer its position. Furthermore, we can find the position of the remain value 3 at position 0 through the 4 command. It can be possible to make the list empty or to remove all the data in the list by 5 command. Now we can test if the list is full or empty. Finally, we can also fill the list randomly, we tell the computer to create a list for us. In our example we got a randomly filled list for 84,87 and 78.

## \*The Stack Type

To start, the user will specify the length of the stack. For this example we select 5. The main point of this implementation is the property of the stack. The last in is the forst out. We can see here we insert with the command tree items 45, 34 and 234. When we print 5, the last item will be pop out. Then it will remain only the first numbers 45 and 34. We can also test if the stack is full or not.

## \*The QueueType

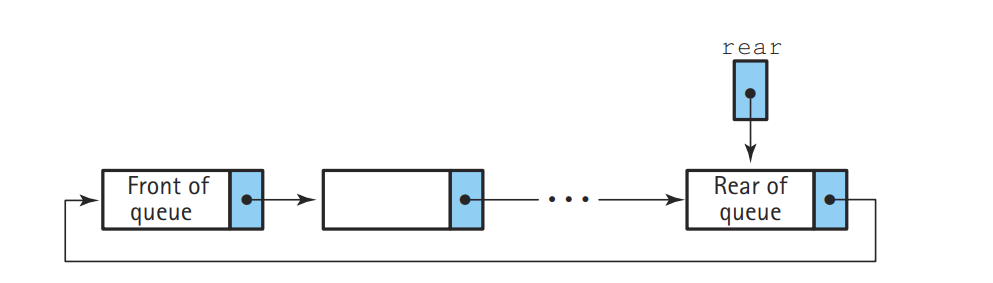
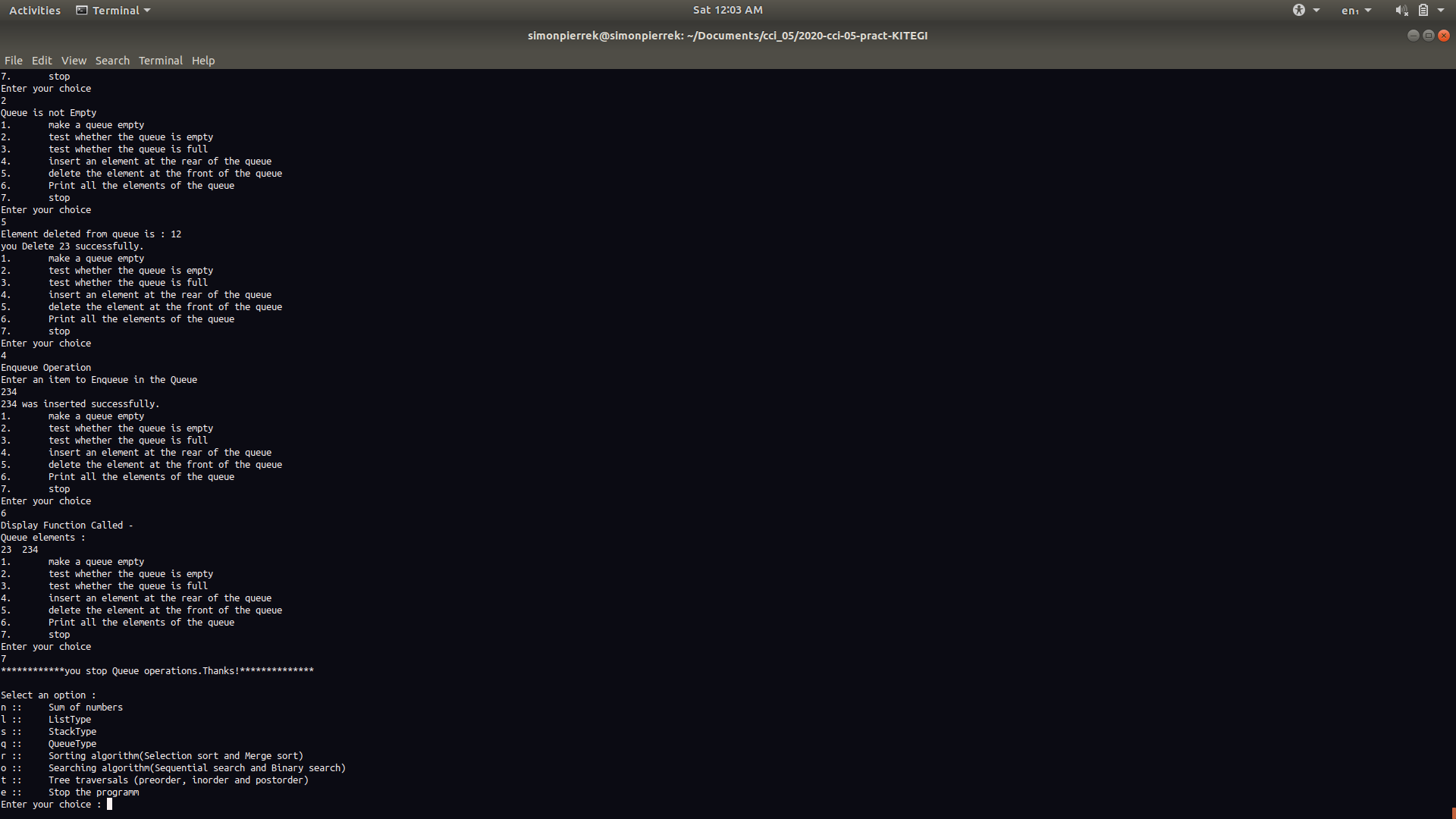
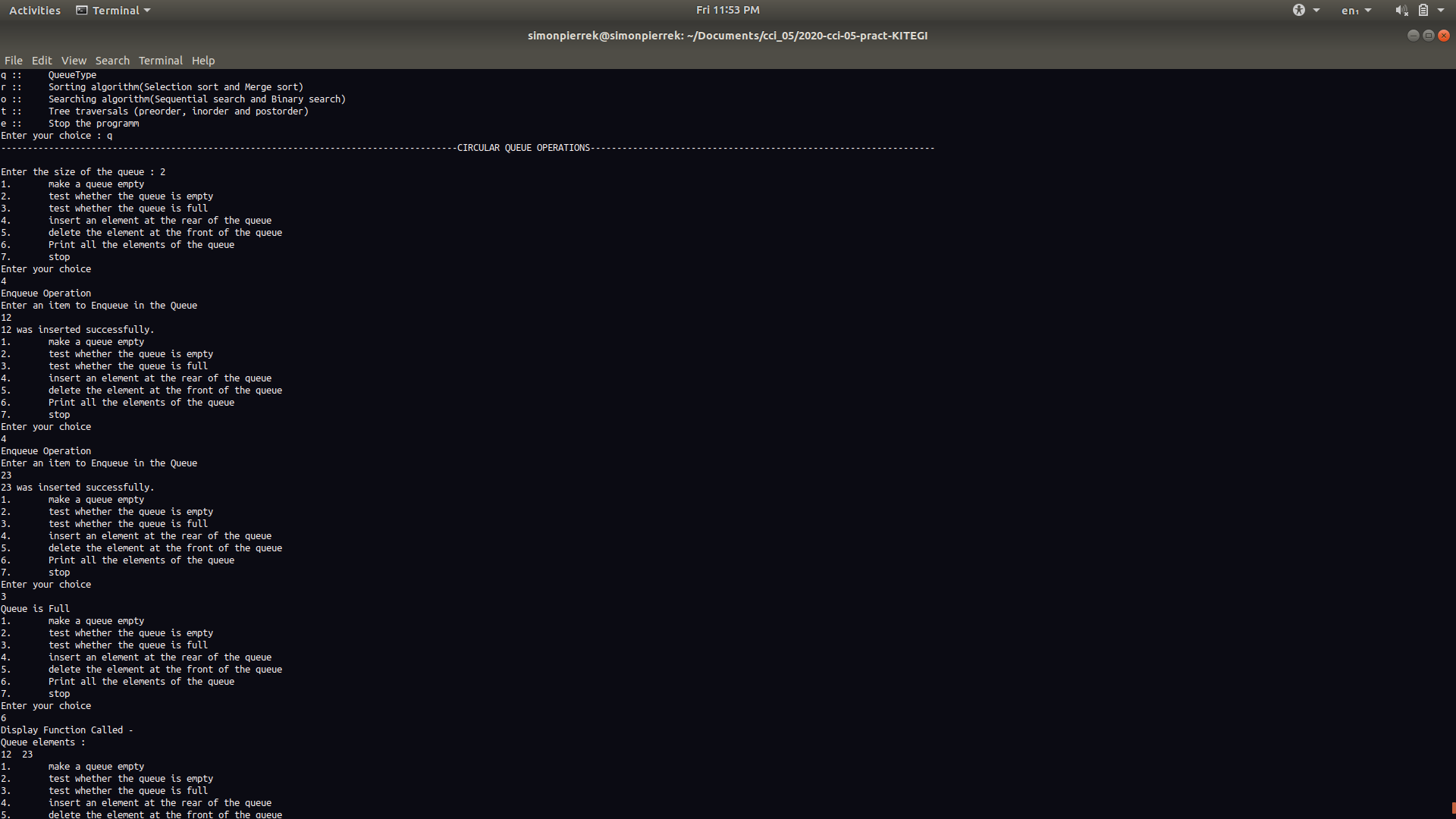


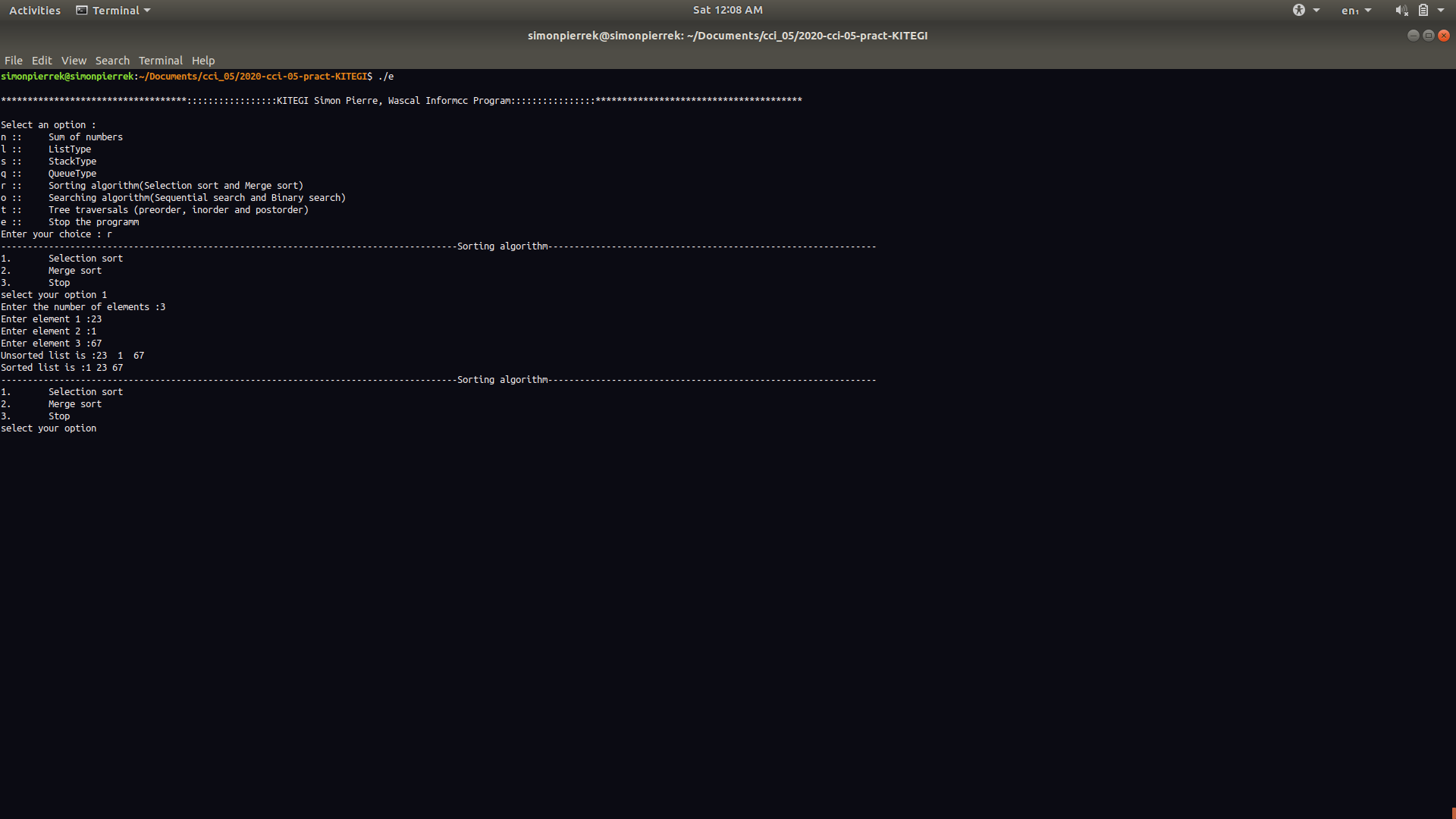
Figure 1: Circular queue representation

In the queue we will specify the size. In our example we choose to have only 2 items. In this particular queue, circular queue. Its property it has no end and we insert by the rear and we delete by the front. In this example we have 2 items at the beginning, 23 and 234. We delete 234, then it remain only 23. We add 12 and we can see that the 12 comes aat the front of the queue. In the module we can test if the queue is full or not.

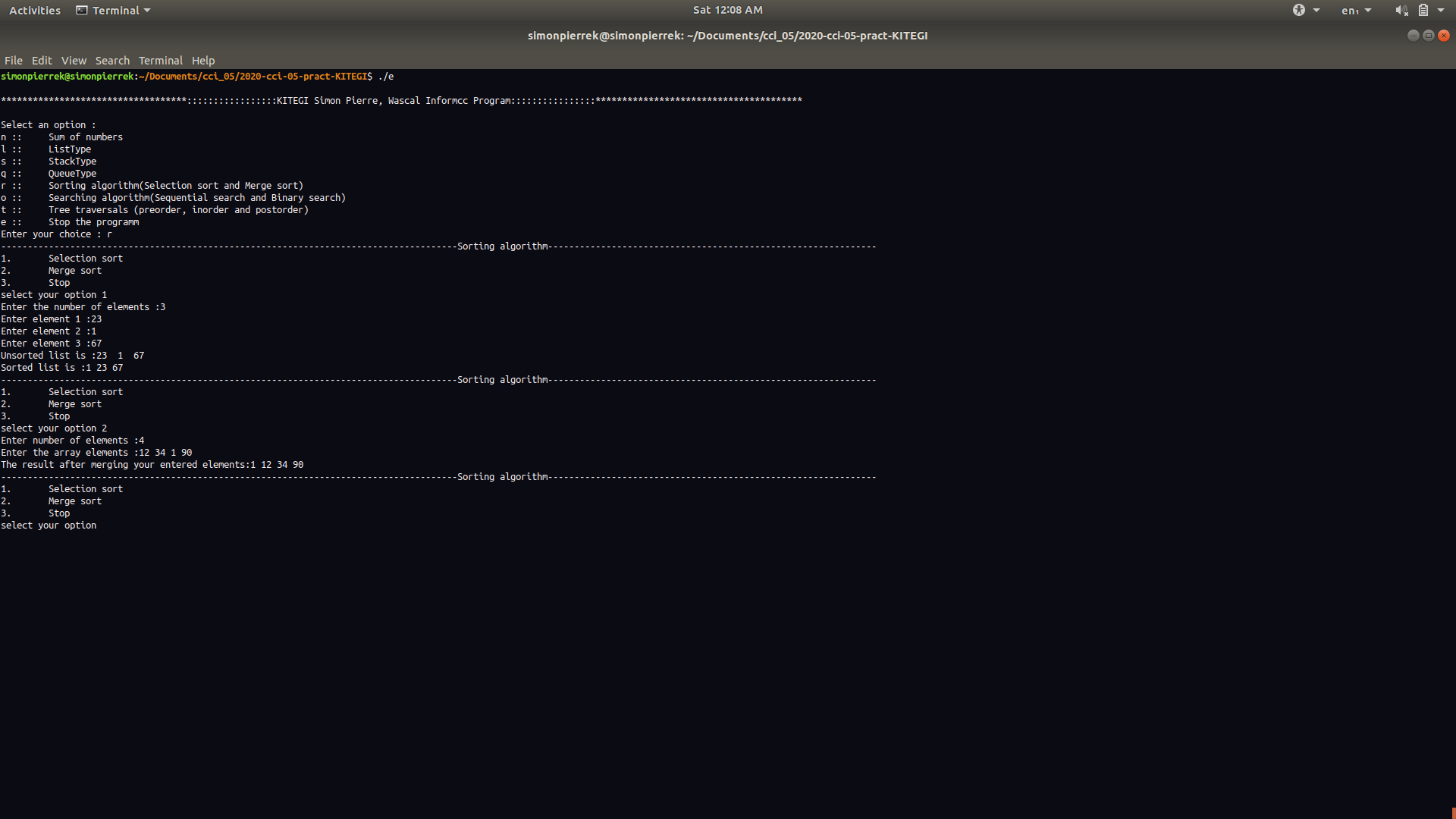
## \*Sorting algorithm

This algorithm is implement to sort all array numbers in ascending form. In this ecample we insert 23, 1 and 67 and us we can see the result is 1, 23 and 67.



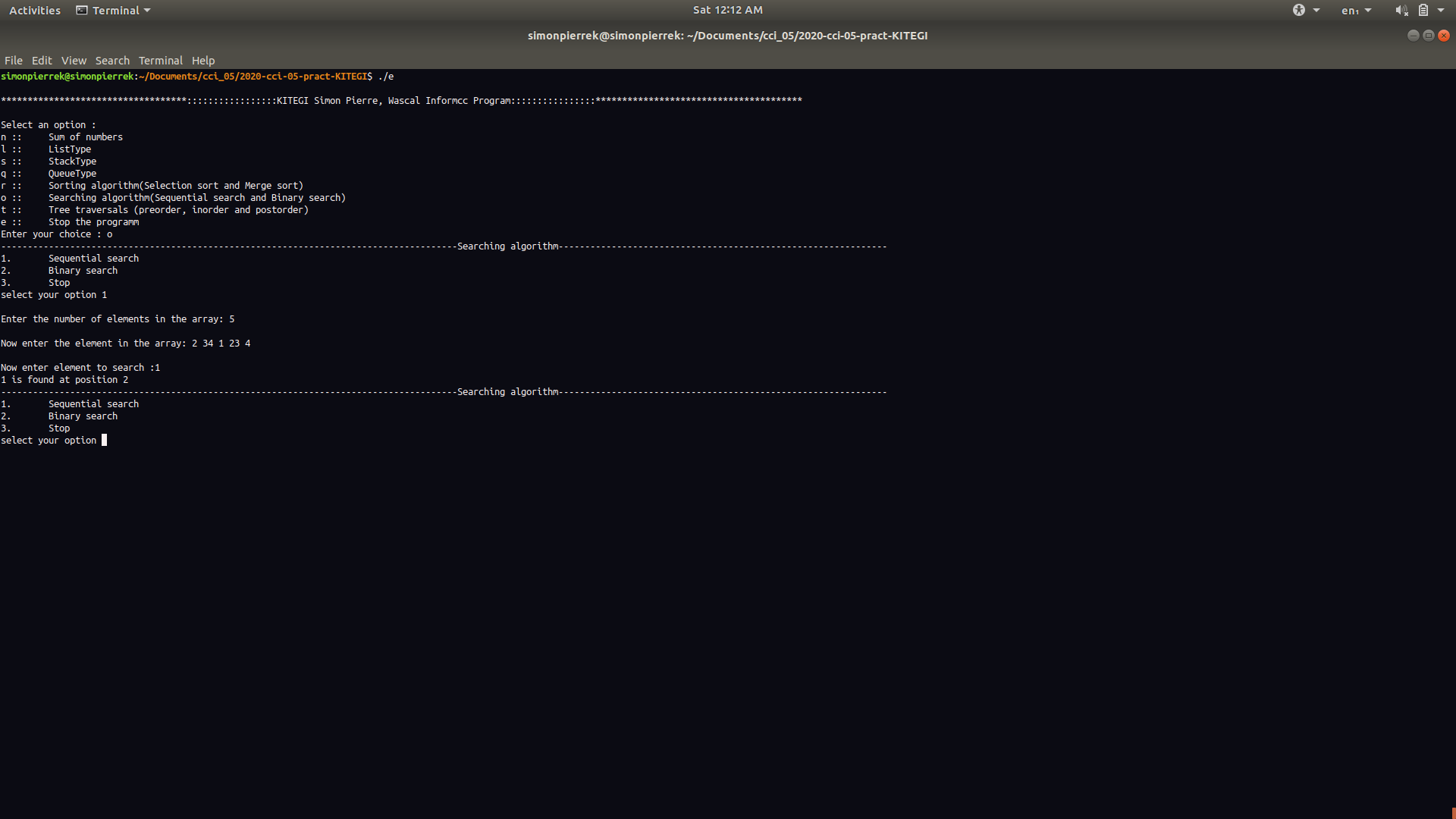
## \*Merging sort

This algorithm divide the array into two halves. It sort the first half with the approach divide and conquer. It does the same with the second. And finaly they put the two sorted halves together with the same approch.



## \*Sequential search

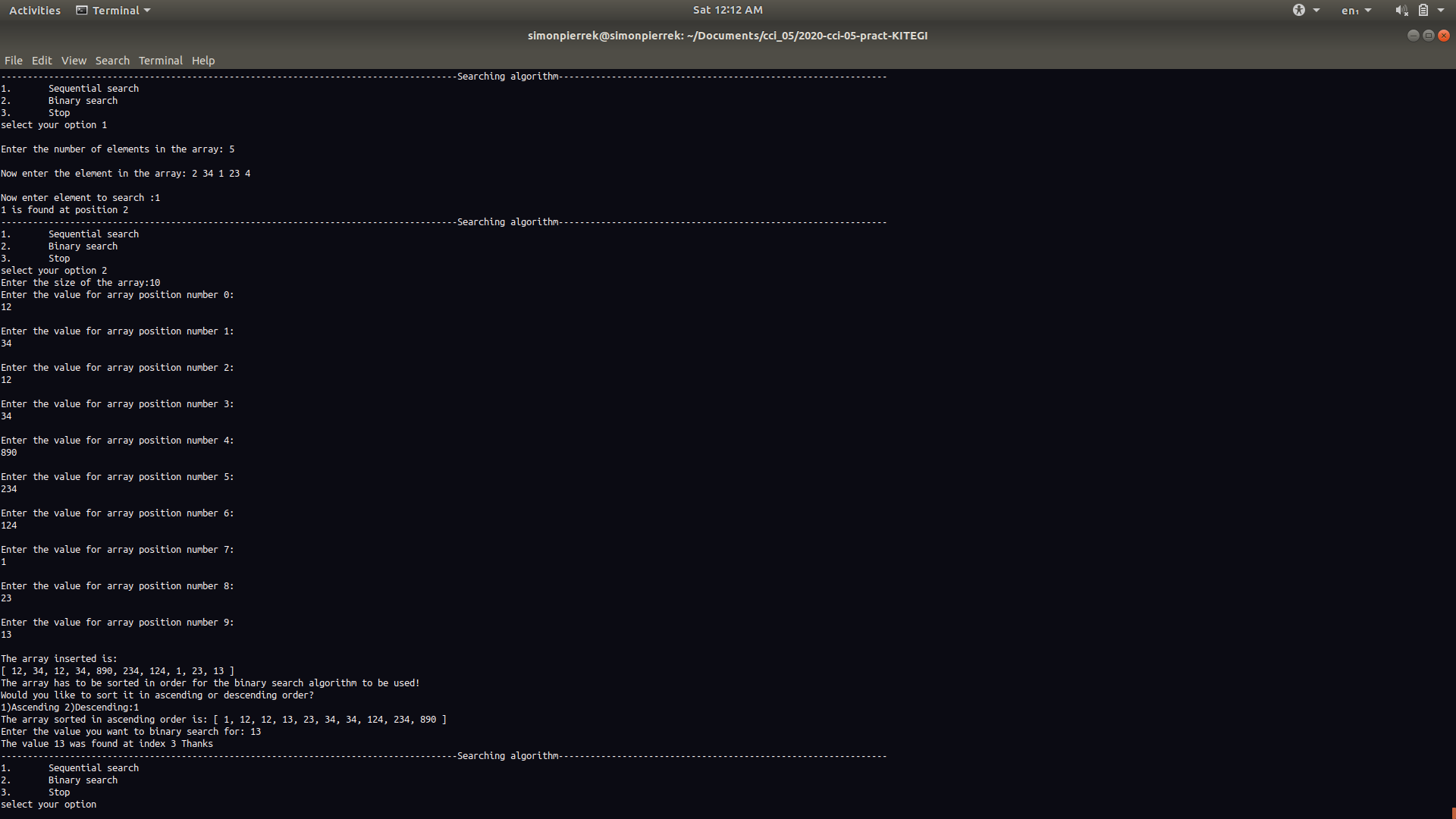
This algorithm is implement to find a specific position of an item in a list. Here we look for the position of 1 in a given list 2, 34, 1, 23, and 4. And we find it at position 2.



## \*Binary search

This algorithm work only for sorted list. Then when the user enter his array, the program will ask him to sort it wheter in ascending or descending sort. After that he can find easily the position of his item in his first list entered.

Here is a specific example of it



## \*Tree traversals

This program is built to represent the data following the principles of binary tree. The program will put the next values after the nodes at right or left. And after we can print the inorder, postorder, and preorder values. As example we have this:

