Exp-06: Familiarization with Standard Template Library - STL (i.e. vector, queue, stack, etc.)

a) Objectives:

- Understand the basics of STL
- Familiarize with vector, queue, stack
- Infix to postfix conversion using stack

b) Prerequisites:

- Functions
- Recursion

c) Theory:

STL: Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc.

STL has four components:

1. Algorithms: sorting, searching. (e.g. binary_search())
2. Containers: vector, queue, stack
3. Functions: functors (e.g. sort(ara, ara+n, comparator)
4. Iterators: for traversing container (e.g. vector<int>::iterator it;)

Task1:

Description:	Store N elements into vector then check whether element X is present in the vector or not.

Sample Input:	N = 5
	Elements = {4, 7, -2, 3, 1}
	X = 3

Sample Output:	Found X

Task2:

Description:	Construct stack / queue	
Sample Input:		
Sample Output:		

Task3:

Description:	Infix to postfix conversion using stack.
Sample Input:	Expression: (2+3)-5*(4/2)

Sample Output:	Output: 23+542/*-

Algorithm to convert Infix To Postfix

Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression Y.

```
1. Push "("onto Stack, and add ")" to the end of X.
2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack
   is empty.
3. If an operand is encountered, add it to Y.
4. If a left parenthesis is encountered, push it onto Stack.
5. If an operator is encountered, then:
    1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack)
       which has the same precedence as or higher precedence than operator.
    2. Add operator to Stack.
       [End of If]
6. If a right parenthesis is encountered, then:
    1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack)
       until a left parenthesis is encountered.
    2. Remove the left Parenthesis.
       [End of If]
       [End of If]
7. END.
```

d) Discussion:

- Round-robin CPU scheduling

e) Homework:

Implement the following tasks on your own. You can discuss with others but copy/pasting code from any source is strictly prohibited. Violation of this rule will result in permanent failure of this course.

1) Perform round-robin CPU scheduling using queue. (Assume at the start there are N process, each requires T execution time in total. Give each program C amount of execution time at each iteration)
2) Evaluate an infix expression using stack. (See book for details)