Lab Assignment 2: Stacks and Queues

1 Parentheses Checker

1.1 Introduction

Stack is a perfect data structure to check the validity of parentheses in arithmetic expression. In this lab task you implement parentheses checker using stack. Given a string input containing only the characters '(', ')', '[', ']' '{' and '}', determine if the input string is balanced. Print "Balanced" if the string is balanced otherwise print "Not Balanced". An input string is balanced if:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.

Additional Constraints:

- Curved ( ) brackets can contain only ( ) brackets.
- Square brackets [ ] can contain only [ ] and ( ) brackets.
- Curly { } brackets can contain { }, [ ] and ( ) brackets.

1.2 Requirements

The stack should be implemented using an array. Once you've got your program implemented correctly, you should be able to produce output like the example shown in Fig 1, where the size of the stack is set to 10. You can choose different value for the size. This depends on how long the input string you allow user to input.

```
mactanhe-1912:Lab 2 tanhe$ ./a.out
Enter an parenthese string: {()()}
Balanced!
mactanhe-1912:Lab 2 tanhe$ ./a.out
Enter an parenthese string: [(){}]
Not Balanced!
mactanhe-1912:Lab 2 tanhe$ ./a.out
Enter an parenthese string: [[[[[[[[[()]]]
Stack Overflow
mactanhe-1912:Lab 2 tanhe$ ./a.out
Enter an parenthese string: [(()]
Not Balanced!
```

Figure 1: example output from the checker

1.3 Deliverables

The deliverable is a source code file in which the checker described above is implemented. Variables in your code should have proper names, and the code has to include sufficient comments for readability.

2 Printer Simulation

2.1 Introduction

Access to shared resources in computer systems are often controlled by a queueing mechanism. A common example is printers. In this lab task you will simulate the scenario of a laboratory printer (as shown in Fig 2). On average there will be one print task in N seconds. The length of the print tasks ranges from 1 to M pages. The printer in the lab can process P pages per minute (i.e. 60 seconds) at good quality.

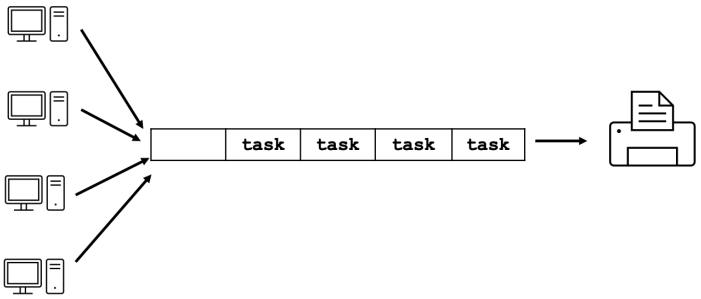

Figure 2: laboratory printer scenario

The simulation runs roughly as follows:

- Set the printer to empty.
- Create a queue of print tasks.
- While (haven't reached clock limit)
  - Does a new print task get created? If so, add it to the queue with the current second as the timestamp upon its arrival.
  - If the printer is not busy and if a task is waiting, remove the next task from the print queue and assign it to the printer.
  - The printer now does one second of printing if necessary, i.e., subtracts one second from the time required for the task. If the task has been completed, in other words the time required has reached zero, the printer is no longer busy.
- Simulation ends.

2.2 Requirements

You should implement the interfaces for printer ADT, task ADT and queue ADT as defined in the header files (in printer_simulation_headers.zip), and a main function for the simulation process. As defined in the queue ADT interface, the queue should be implemented using linked list. Regarding the main function it is up to you to decide how to code it. Once you've got your program implemented correctly, you should be able to produce output like the example shown in Fig 3, where $N = 2$, $M = 5$, $P = 40$ and the simulation runs for 10 seconds. You can choose different values for the simulation parameters in your program.

```
mactanhe-1912:printer_simulation tanhe$ ./main

THE PRINTER IS NO LONGER BUSY
TASKS IN QUEUE:
TASK [arrives at 1 second, 5 pages]

THE PRINTER IS BUSY
6 seconds to complete the current task.
TASKS IN QUEUE:
TASK [arrives at 2 second, 4 pages]

THE PRINTER IS BUSY
5 seconds to complete the current task.
TASKS IN QUEUE:
TASK [arrives at 2 second, 4 pages]

THE PRINTER IS BUSY
4 seconds to complete the current task.
TASKS IN QUEUE:
TASK [arrives at 2 second, 4 pages]

THE PRINTER IS BUSY
3 seconds to complete the current task.
TASKS IN QUEUE:
TASK [arrives at 2 second, 4 pages]

THE PRINTER IS BUSY
2 seconds to complete the current task.
TASKS IN QUEUE:
TASK [arrives at 2 second, 4 pages]

THE PRINTER IS BUSY
1 seconds to complete the current task.
TASKS IN QUEUE:
TASK [arrives at 2 second, 4 pages]
TASK [arrives at 7 second, 5 pages]

THE PRINTER IS NO LONGER BUSY
TASKS IN QUEUE:
TASK [arrives at 2 second, 4 pages]
TASK [arrives at 7 second, 5 pages]

THE PRINTER IS BUSY
5 seconds to complete the current task.
TASKS IN QUEUE:
TASK [arrives at 7 second, 5 pages]
TASK [arrives at 9 second, 3 pages]

SIMULATION ENDS
```

Figure 3: example program output

2.3 Deliverables

The deliverable is a zip file in which the files are organized as in Fig 4. Variables in your code files should have proper names, and the code has to include sufficient comments for readability.

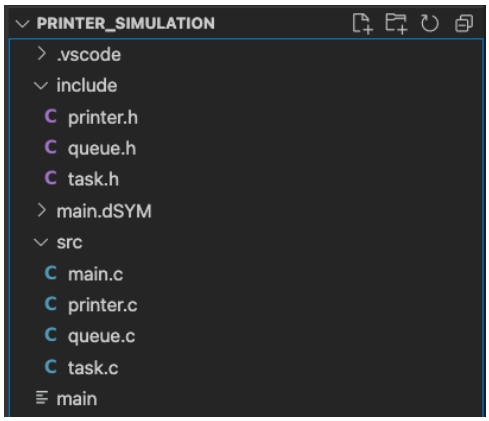

Figure 4: file organization