==========

数据结构与算法大作业：日志分析系统

3 周，100 分

==========

一、问题描述

实现命令行程序，读取服务器日志文件，完成以下 5 个功能：

1. 存储日志（用双向链表保存所有日志）
2. 时间窗口查询（循环队列）
3. 关键词搜索（用 KMP 在日志消息中查找关键词）
4. 模块错误统计（用二叉搜索树统计各模块 ERROR 数量）
5. 操作撤销（支持按行号删除日志，并用栈实现撤销最近的删除）

二、日志文件格式

每行一条日志，格式固定：

[时间戳] [级别] [模块] 消息内容

时间戳格式：YYYY-MM-DD HH:MM:SS

级别：INFO, WARN, ERROR

模块：任意字符串（无空格）

消息内容：任意字符串（可含空格）

示例：

[2025-01-15 10:32:45] ERROR Database Connection timeout

[2025-01-15 10:32:46] INFO Network Request received

[2025-01-15 10:32:47] ERROR Database Query failed

[2025-01-15 10:32:48] WARN Auth Token expired

三、数据结构要求

禁止使用 STL 容器（list, queue, stack, map, set），必须手写：

1. 双向链表：存储所有日志
   - 节点包含：时间戳、级别、模块、消息、前驱指针、后继指针
   - 支持：头插、尾插、删除指定行号（行号从 1 开始）、从头到尾/从尾到头遍历。

2. 循环队列：固定大小 1000，存储最近 1000 条日志指针
   -定长 1000；存放“指向链表节点的指针”；
   -新日志入队：未满：正常入队；已满：覆盖最旧元素；
   -支持遍历，从队头到队尾按时间从旧到新

3. 二叉搜索树：key=模块名，value=ERROR 计数
   - 只统计 ERROR 级别
   - 支持：插入、查找、中序遍历

4. 栈：存储最近 5 次操作（用于撤销）
   - 存储整个链表状态（深拷贝或快照）
   - 支持：压栈、出栈

5. KMP 算法：在消息内容中查找关键词
   - 必须手写 next 数组计算
   - 不允许使用 string::find()

四、命令格式

程序运行后进入交互模式，支持以下命令（命令不区分大小写）：

命令 1：LOAD <文件名>
功能：读取日志文件，构建链表、循环队列、BST
输出：Loaded <数量> entries

命令 2：FILTER <开始时间> <结束时间>
功能：从链表中查找时间在 [开始时间, 结束时间] 闭区间内的所有日志并输出；命令中的时间格式为 YYYY-MM-DD_HH:MM:SS（日期和时间之间用下划线）。
输出：每行原样打印日志；如果没有结果则不输出任何日志行。

命令 3：SEARCH <关键词>
功能：使用 KMP 算法在消息内容中搜索关键词（大小写敏感）
输出：匹配的日志数量和每条日志原文

命令 4：STATS
功能：按模块输出 ERROR 数量，排序从多到少；数量相同按模块名字典序；
输出格式：
模块名 1: <数量> errors
模块名 2: <数量> errors
Database: 3 errors
Network: 2 errors
...

命令 5：DELETE <行号>
功能：删除指定行号的日志（行号从 1 开始），同时：（1）从循环队列中移除对应节点指针（如果存在）；（2）更新 BST 中该模块的 ERROR 统计（如果是 ERROR 级别）；（3）在删除前，将当前状态作为快照压入栈（最多保存最近 5 次）。

输出：Deleted entry <行号>，注意：删除后，后面的日志行号会重新编号。

命令 6：UNDO
功能：撤销上一次 DELETE 操作，将整个系统状态恢复到那次 DELETE 之前。恢复链表、循环队列、BST；最多撤销 5 次；只能撤销 DELETE 操作，不能撤销 LOAD / SEARCH / FILTER。
输出：若成功撤销为：Undo successful；
若栈为空：No more undo。

命令 7：RECENT <数量>
功能：输出循环队列中最近 N 条日志（N<=1000）
输出：每行一条日志原文

命令 8：EXIT
功能：退出程序

五、测试用例

【用例 1：基本功能】15 分
输入文件：test1.txt（10 行）
[2025-01-15 10:00:00] INFO Network Started
[2025-01-15 10:00:01] ERROR Database Connection failed
[2025-01-15 10:00:02] WARN Auth Token missing
[2025-01-15 10:00:03] ERROR Database Timeout
[2025-01-15 10:00:04] INFO Network Request OK
[2025-01-15 10:00:05] ERROR Network Socket closed
[2025-01-15 10:00:06] ERROR Database Deadlock
[2025-01-15 10:00:07] INFO Auth Login success
[2025-01-15 10:00:08] ERROR Network Packet lost
[2025-01-15 10:00:09] INFO Database Query OK

命令序列：
> LOAD test1.txt
期望输出：Loaded 10 entries
评分：正确输出 3 分，否则 0 分

> STATS
期望输出（顺序必须匹配）：
Database: 3 errors
Network: 2 errors
评分：完全正确 3 分，部分正确 1 分，错误 0 分

> SEARCH failed

期望输出：
Found 1 match(es)
[2025-01-15 10:00:01] ERROR Database Connection failed
评分：正确输出 3 分，数量对但内容错 2 分，否则 0 分

> FILTER 2025-01-15_10:00:02 2025-01-15_10:00:04
期望输出：
[2025-01-15 10:00:02] WARN Auth Token missing
[2025-01-15 10:00:03] ERROR Database Timeout
[2025-01-15 10:00:04] INFO Network Request OK
评分：完全正确 3 分，部分正确 1 分，否则 0 分

> RECENT 3
期望输出：
[2025-01-15 10:00:07] INFO Auth Login success
[2025-01-15 10:00:08] ERROR Network Packet lost
[2025-01-15 10:00:09] INFO Database Query OK
评分：完全正确 3 分，部分正确 1 分，否则 0 分

【用例 2：DELETE 和 UNDO】20 分
接上一用例，继续输入：

> DELETE 2
期望输出：Deleted entry 2
评分：正确输出 2 分

> STATS
期望输出：
Database: 2 errors
Network: 2 errors
评分：正确 3 分（验证 BST 更新），否则 0 分

> UNDO
期望输出：Undo successful
评分：正确输出 2 分

> STATS
期望输出：
Database: 3 errors
Network: 2 errors
评分：正确 5 分（验证状态恢复），部分正确 2 分，否则 0 分

> DELETE 1

> DELETE 3
> DELETE 5
> DELETE 7
> DELETE 9
> UNDO
> UNDO
> UNDO
> UNDO
> UNDO
> UNDO
期望输出：前 5 次 Undo 输出"Undo successful"，第 6 次输出"No more undo"
评分：完全正确 8 分，部分正确 4 分，否则 0 分

【用例 3：大文件与性能】20 分
输入文件：test3.txt（50000 行，约 5MB）
时间范围：2025-01-01 00:00:00 到 2025-01-31 23:59:59
模块：Database, Network, Auth, Cache, Storage（随机分布）
级别：INFO 60%, WARN 25%, ERROR 15%

命令序列：
> LOAD test3.txt
要求：完成时间 < 3 秒，内存使用 < 200MB
评分：时间和内存都满足 10 分，仅时间满足 5 分，都不满足 0 分

> SEARCH timeout
要求：KMP 搜索完成时间 < 1 秒
评分：满足 5 分，否则 0 分

> RECENT 1000
要求：输出 1000 行，时间 < 0.5 秒
评分：满足 5 分，否则 0 分

【用例 4：边界情况】15 分

4.1 空文件（3 分）
输入：empty.txt（0 行）
> LOAD empty.txt
期望输出：Loaded 0 entries
> STATS
期望输出：（无输出）
> SEARCH test
期望输出：Found 0 match(es)

4.2 单行文件（3 分）
输入：single.txt（1 行）
[2025-01-15 10:00:00] ERROR Test Single line
> LOAD single.txt
> DELETE 1
> UNDO
> STATS
期望输出：Test: 1 errors

4.3 循环队列溢出（3 分）
输入：large.txt（2000 行）
> LOAD large.txt
> RECENT 1000
期望输出：最后 1000 行（行号 1001-2000）

4.4 无匹配搜索（3 分）
> SEARCH nonexistent_keyword_12345
期望输出：Found 0 match(es)

4.5 时间范围无结果（3 分）
> FILTER 1999-01-01_00:00:00 1999-12-31_23:59:59
期望输出：（无输出）

【用例 5：KMP 正确性】15 分
输入文件：kmp_test.txt
[2025-01-15 10:00:00] INFO Test ababcababa
[2025-01-15 10:00:01] INFO Test abababab
[2025-01-15 10:00:02] INFO Test aaaaa
[2025-01-15 10:00:03] INFO Test abcabc

> SEARCH ababa
期望输出：
Found 2 match(es)
[2025-01-15 10:00:00] INFO Test ababcababa
[2025-01-15 10:00:01] INFO Test abababab
评分：完全正确 5 分，部分正确 2 分

> SEARCH aaa
期望输出：
Found 1 match(es)
[2025-01-15 10:00:02] INFO Test aaaaa
评分：5 分

> SEARCH abc
期望输出：
Found 2 match(es)
[2025-01-15 10:00:00] INFO Test ababcababa
[2025-01-15 10:00:03] INFO Test abcabc
评分：5 分

【用例 6：状态一致性】15 分
测试 DELETE 后各数据结构的一致性

输入：consistency.txt（20 行，前 10 行都是 ERROR Database）
> LOAD consistency.txt
> DELETE 5
> FILTER 2025-01-15_10:00:00 2025-01-15_23:59:59
验证：输出 19 行（不含第 5 行）（5 分）

> STATS
验证：Database: 9 errors（BST 正确更新）（5 分）

> RECENT 20
验证：最后 19 行（循环队列正确更新）（5 分）

六、评分标准

总分 100 分 = 自动测试 90 分 + 代码质量 10 分

【自动测试 90 分】
- 用例 1：基本功能（15 分）
- 用例 2：DELETE 和 UNDO（20 分）
- 用例 3：大文件与性能（20 分）
- 用例 4：边界情况（15 分）
- 用例 5：KMP 正确性（15 分）
- 用例 6：状态一致性（15 分）
  注：超过 100 分的部分作为 buffer

【代码质量 10 分】
1. 注释完整（2 分）：每个函数有功能说明
2. 无内存泄漏（3 分）：valgrind 检测通过
3. 代码结构清晰（2 分）：合理的文件组织
4. 变量命名规范（1 分）：有意义的命名
5. 边界检查（2 分）：数组越界、空指针检查

【扣分项】
- 使用 STL 容器（list, queue, stack, map, set）：-30 分
- 使用 string::find()等库函数而非 KMP：-10 分
- 编译错误：0 分
- 段错误/崩溃：该用例 0 分
- 迟交每天：-5 分

七、提交内容

提交 zip 文件，包含：

1. 源代码（必需）
   - main.cpp
   - LinkedList.h, LinkedList.cpp
   - CircularQueue.h, CircularQueue.cpp
   - BST.h, BST.cpp
   - UndoStack.h, UndoStack.cpp
   - KMP.h, KMP.cpp
   - Makefile（生成可执行文件"loganalyzer"）

2. README.txt（必需）
   内容包括：
   - 编译命令
   - 运行方法
   - 数据结构设计简述（每个 300 字以内）
   - 已知问题说明

3. 学号.txt（必需）
   仅包含学号一行，用于生成个性化测试数据

八、编译与运行

编译：
$ make
生成可执行文件：loganalyzer

运行：
$ ./loganalyzer
进入交互模式，等待输入命令

测试：

$ ./loganalyzer < test_commands.txt
从文件读取命令序列

九、注意事项

---

1. 时间格式：YYYY-MM-DD HH:MM:SS（有空格），命令中用下划线：YYYY-MM-DD_HH:MM:SS
2. 行号从 1 开始，不是 0
3. STATS 输出按 ERROR 数量从高到低，数量相同时按模块名字典序
4. SEARCH 大小写敏感，完全匹配子串
5. FILTER 包含起止时间（闭区间）
6. RECENT 输出顺序：从旧到新（队头到队尾）
7. UNDO 只能撤销 DELETE 操作，其他操作不可撤销
8. 循环队列固定 1000，超出覆盖最旧的
9. BST 只统计 ERROR，不统计 INFO 和 WARN
10. 所有输出末尾有换行符

十、时间安排建议

第 1 周：
- Day 1-3：实现链表、循环队列、BST（独立测试）
- Day 4-5：实现 LOAD、FILTER、STATS 命令
- Day 6-7：测试用例 1

第 2 周：
- Day 1-3：实现 KMP 算法（手写 next 数组）
- Day 4-5：实现 SEARCH 和 RECENT 命令
- Day 6-7：测试用例 5

第 3 周：
- Day 1-3：实现 DELETE、UNDO（栈+状态恢复）
- Day 4-5：测试用例 2、6（调试状态一致性）
- Day 6：性能优化，测试用例 3
- Day 7：边界测试用例 4，整理提交

十一、个性化测试数据生成（基于学号）

为保证学习效果，课程将检查多结构状态一致性，通过代码结构、注释风格、与课堂练习和平时作业的一致性，综合判断是否存在严重依赖大模型的情况。一旦认定存在严重依赖大模型或抄袭行为，该大作业成绩可能被记为 0 分；你只需要保证：你的程序对于任何满足日志格式的文件，都能正确执行上述命令。

请务必自己思考、自己调试，这才是本次大作业真正的意义。

个性化测试：
提交学号后，系统自动生成 3 个测试文件：
- personal_test1.txt（100 行）：10 个特定查询任务
- personal_test2.txt（5000 行）：性能测试
- personal_test3.txt（随机复杂场景）

个性化任务示例（基于学号 seed=12345）：
Task 1: FILTER 2025-01-15_14:23:10 2025-01-15_16:45:30，预期输出 23 行
Task 2: SEARCH database_connection_pool，预期输出 7 行
Task 3: 执行 DELETE 15, DELETE 27, UNDO, UNDO 后 STATS，预期 Database: 18 errors
...共 10 个任务，每个 2 分，共 20 分计入总分
个性化测试在提交后 48 小时内返回结果。