기존 regex.cpp 코드에서 추가적인 멤버 변수와 함수 등을 작성하였다.

추가된 멤버 변수)

loop : 반복의 여부(ex. +, \*, {m, n}), 0일 경우, 반복하지 않는 노드이고, 1일 경우, 반복하는 노드이다.

start : 반복의 최소 횟수를 의미한다.

end : 반복의 최대 횟수를 의미한다.

parent : 현재 노드의 부모(next 멤버 변수에 현재 노드가 들어가 있는 노드)를 의미한다.

cnt : 반복의 횟수를 기록하는 멤버 변수이다.

추가된 함수)

char getmatch() : 현재 노드의 match 값을 반환하는 함수이다.

void setpt(int num) : 현재 노드의 parent 값을 num으로 설정하는 함수이다.

void setcnt(int n) : 현재 노드의 start 값을 n으로 지정하고, end값을 int의 최댓값으로 설정한다.

void setcnt(int a, int b) : 현재 노드의 start 값을 a로 지정하고, end값을 b로 지정한다.

void setloop() : 현재 노드의 loop값을 1로 지정한다.

int check(char alpha) : 매개변수로 입력된 alpha가 대소문자 알파벳에 해당하면 0, 정규표현식에 쓰이는 특수 기호이면 1, 그외의 경우면 -1을 반환한다.

변경된 함수)

void init(const string &\_name, bool \_isTerminal, char \_match) : 현재 노드의 loop와 start, end, cnt값을 0으로 각각 지정하는 구문을 추가했다.

void input(const char &ch) : 만약 현재 노드가 \*나 +와 같은 것들이 붙어서 반복하는 노드인 경우, 현재 노드의 state가 계속 활성화(1)되어 있을 필요가 있다. 그렇기에 start와 end, cnt를 활용하여 코드를 작성…해보려 했지만 시간이 촉박해서 하지 못했습니다…

void test(vector<node &s, const string &str) : 기존에 존재하던 s[0].transition().activation() 구문은 abc|def와 같은 정규표현식 처리기에서는 시작 노드가 a와 d로 서로 다르기 때문에 사용할 수 없다. 그러므로 s[i].getpt()를 이용하여 부모 노드가 s0인 모든 노드에 대해 for문으로 s[i].transition().activation() 구문을 실행하도록 변경했다. 이를 위해 main 함수에서 s[0].init(“s0”, 0, ‘@’) 구문을 작성하여 root 노드를 미리 만들어 두었다.

main 함수 내 변경사항)

주어진 범위 내에서 입력을 할 수 있도록 하기 위해, check 함수를 이용하여 반환값이 -1인 경우에 “invalid\_syntax”가 출력된 후에 프로그램을 즉시 종료하도록 했다. 또한 정규표현식의 노드들을 일일이 만들고 연결하던 기존 코드를 수정하여 처음에 regex 변수에 정규표현식을 입력받을 수 있게 했다. 입력받은 regex 값은 sstream 라이브러리를 활용해서 “|”를 기준으로 split하여 알파벳, “.”, “+”, “\*”, “{m, n}”에 따라 노드 생성 및 연결을 자동화…시키려 했지만 현재는 “.”과 “|”에만 반응한다. 단어의 반복의 처리에 있어서는 반복의 끝을 처리할 수 있어야 하는데, 그 방법에 대한 아이디어가 잘 떠오르지 않았다. 예를 들어, 정규표현식 “ab.\*cd”가 입력된 경우에 “abababcdcdcd”와 같은 문자열을 처리할 때, “abababcdcdcd”를 인식한다. 하지만 노드는 연결된 부분을 순차적으로 따라가기 때문에 ”abababcd”를 인식한 후에 멈춰서는 문제점이 있다. 이를 해결할 방법이 주어진 시간 내에 떠오르지 않아서 하지 못했습니다… 더 열심히 공부해오겠습니다…