# MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY


## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**COURSE CODE**: CSE-303

**COURSE NAME: Compiler**

**Term Paper**

| Student ID | Name |
|---|---|
| 202214049 | Md. Nahul Rahman |

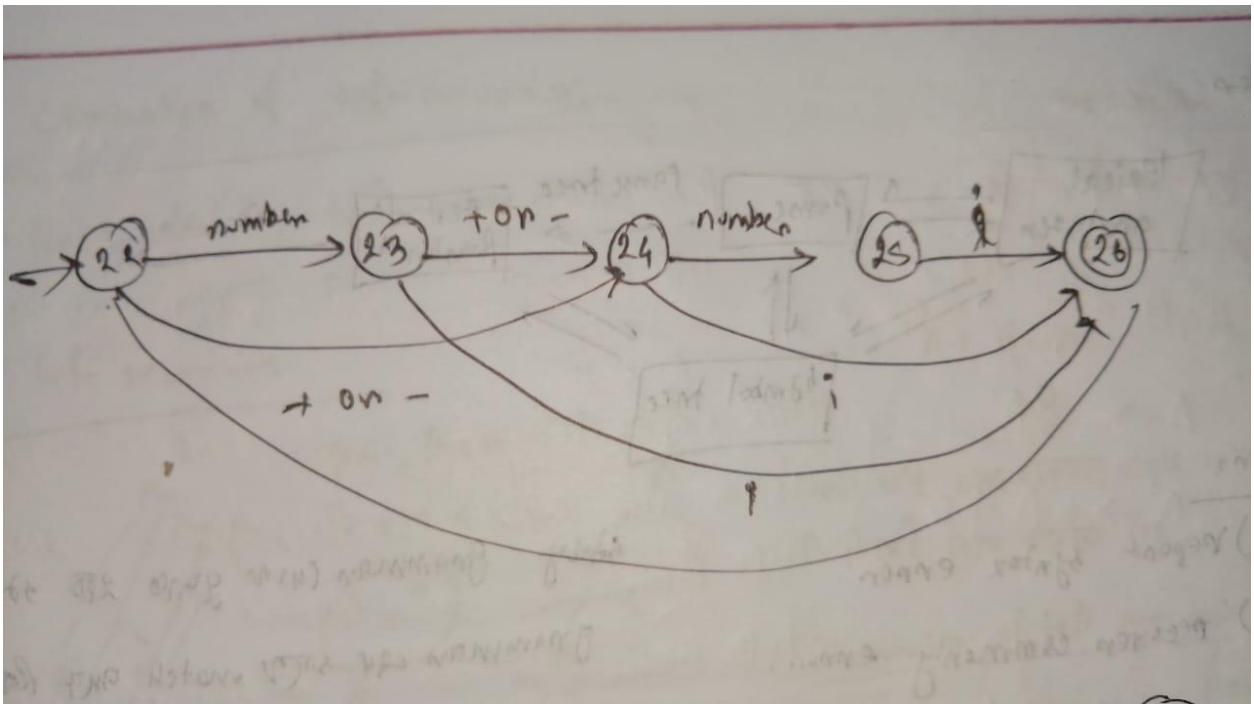
Pattern for complex number

digits -> [0-9]+

numbers -> digits(.digits)? ([Ee][+-]?digits)

complex -> numbers(+|-)(numbers)?i|[+-]?number?i

State transition diagram for complex numbers



Code for Complex number

```
#include <bits/stdc++.h>
using namespace std;

FILE *in;
FILE *out;

class TOKEN {
public:
    string token_name;
    string lexeme;
};

void retract() {
    fseek(in, -1, SEEK_CUR);
}

TOKEN getComplexNumber() {
    TOKEN retToken;
    char c;
    string lexeme = "";
    int state = 0;
    while (1) {
        c = fgetc(in);
        if (c == EOF) {
            break;
        }

        switch (state) {
            case 0:
                if (isdigit(c)) {
                    state = 1;
                    lexeme += c;
                }
                else if(c == '\n') continue;
                else if (c == '+' || c == '-') {
                    lexeme += c;
```

```
                state = 2;
            } else if (c == 'i') {
                lexeme += c;
                state = 4;
            }
            else {
                lexeme += c;
                state = 5;
            }

            break;

        case 1:
            if (isdigit(c)) {
                lexeme += c;
            } else if (c == '+' || c == '-') {
                lexeme += c;
                state = 2;
            } else if (c == 'i') {
                lexeme += c;
                state = 4;
            }
            else {
                lexeme += c;
                state = 5;
            }

            break;

        case 2:
            if (isdigit(c)) {
                state = 3;
                lexeme += c;
            } else if (c == 'i') {
                lexeme += c;
                state = 4;
            }
            else {
                lexeme += c;
```

```
                state = 5;
            }

            break;

        case 3:
            if (isdigit(c)) {
                lexeme += c;
            } else if (c == 'i') {
                lexeme += c;
                state = 4;
            }
            else {
                lexeme += c;
                state = 5;
            }

            break;

        case 4:
            retract();
            retToken.token_name = "Complex Number";
            retToken.lexeme = lexeme;
            return retToken;

        case 5:
            retract();
            retToken.token_name = "Invalid";
            retToken.lexeme = lexeme;
            return retToken;
    }
}

if (state == 4) {
    retToken.token_name = "Complex Number";
    retToken.lexeme = lexeme;
} else if (state == 0) {
    retToken.token_name = "";
    retToken.lexeme = "";
```

```
    } else {
        retToken.token_name = "Invalid";
        retToken.lexeme = lexeme;
    }
    return retToken;
}

void init() {
    while (true) {
        TOKEN token = getComplexNumber();
        if (token.token_name.empty()) {
            break;
        }
        if (token.token_name == "Complex Number")
            cout << "<Complex Number, " << token.lexeme <<
">" << endl;

    }
}

int main() {
    in = fopen("input.txt", "r");
    out = freopen("output.txt", "w", stdout);
    init();
    fclose(in);
    fclose(out);
    return 0;
}
```