KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

(CSE)

Compiler Project Proposal Report Manual

Tools: Using Flex & Bison

Submitted To:

Md. Badiuzzaman Shuvo

Lecturer

Department of Department of Computer Science and Engineering (CSE)

Khulna University of Engineering & Technology (KUET)

Subah Nawar

Lecturer

Department of Department of Computer Science and Engineering (CSE)

Khulna University of Engineering & Technology (KUET)

Submitted By:

Adiba Tahsin

Roll No: 2107031

Year: Third

Semester: Second

Department of Computer Science and Engineering (CSE)

Khulna University of Engineering & Technology (KUET)

Date of Submission: January 19, 2026

EmotionScript: A Domain-Specific Language for Modeling Human Emotions and Behavior

Project Introduction:

EmotionScript is a domain-specific programming language designed to model human emotions, behavior, and decision-making using structured programming constructs. Instead of traditional technical keywords, the language introduces human-centric terminology such as thoughts, memories, emotions, and interpretations to represent variables, state, and logic. Although the syntax is expressive and thematic, every construct in EmotionScript maps directly to well-known programming concepts such as variables, functions, conditionals, loops, object-oriented structures, and finite state machines.

This project focuses on designing and implementing the front end of the EmotionScript compiler using Flex for lexical analysis and Bison for syntax analysis. A visual compiler simulator is also included to illustrate the internal stages of compilation, such as tokenization, parsing, syntax tree generation, and semantic checking. The goal is to demonstrate compiler design concepts through a unique yet technically sound programming language.

Keywords Mapping:

SL	EmotionScript Keyword	Equivalent in C	Description
1	mind <name>	int main()	Program entry point
2	awake	start of main	Begins execution
3	sleep	return 0;	Normal termination
4	abort	exit(1)	Abnormal termination
5	scene	{}	Logical block
6	count	int	Integer value
7	measure	float	Real number
8	level	float	Emotional intensity (0–1)

9	truth	bool	Boolean
10	words	char*	String
11	emotion	enum (internal)	Emotional state
12	thought	local variable	Temporary value
13	memory	static/global	Persistent value
14	trait	const	Immutable attribute
15	state	mutable variable	Time-varying state
16	set	=	Assignment
17	speak()	printf()	Output
18	listen()	scanf()	Input
19	alert()	printf()	Warning message
20	##	//	Single-line comment
21	#* … *#	/* … */	Multi-line comment
22	interpret name(params) yields type	function	Function definition
23	reflect expr	return expr;	Return statement
24	invoke name()	function call	Function invocation
SL	**Keyword**	**C Equivalent**	**Description**
25	`if_feel (condition)`	`if`	Conditional check
26	`else_if_feel (condition)`	`else if`	Secondary condition
27	`else_feel`	`else`	Alternative branch
28	`end_feel`	end of if	Close conditional

29	`decide`	if–else chain	Multi-branch decision
30	`otherwise`	`else`	Default branch
SL	**Keyword**	**C Equivalent**	**Description**
31	`ruminate (condition)`	`while`	Loop
32	`replay n`	`for`	Fixed iteration
33	`break_free`	`break`	Exit loop
34	`continue_flow`	`continue`	Skip iteration
SL	**Keyword / Grammar**	**C Equivalent**	**Description**
35	`emotion joy`	enum value	Declare emotion
36	`emotion X composed_of A, B`	boolean rule	Compound emotion
37	`trigger emotion`	function call	Activate emotion
38	`emotion ++ / --`	increment/decrement	Adjust intensity
SL	**Keyword / Grammar**	**C Equivalent**	**Description**
39	`transition A -> B when condition`	state transition	FSM transition
40	`current_emotion`	enum variable	Active emotion
SL	**Keyword**	**C Equivalent**	**Description**
41	`persona`	`struct`	Class-like definition
42	`manifest obj : Type`	struct variable	Object creation
43	`open`	public	Public access
44	`guarded`	private	Private access
45	`shared`	protected	Protected access
46	`evolves_from`	inheritance	Inheritance
47	`reshape`	override	Method override
SL	**Keyword**	**C Equivalent**	**Description**
48	`magnitude(x)`	`abs(x)`	Strength
49	`balance(x)`	`sqrt(x)`	Stability
50	`estimate_up(x)`	`ceil(x)`	Upper estimate
51	`estimate_down(x)`	`floor(x)`	Lower estimate
52	`gap(a,b)`	`abs(a-b)`	Difference
53	`normalize(x)`	clamp	Scale to [0–1]

54	`surge(x)`	nonlinear boost	Increase
SL NO	**Keyword (EmotionScript)**	**Keyword in C**	**Description**
50	sine(x)	sin(x)	Sine trigonometric function
51	cosine(x)	cos(x)	Cosine trigonometric function
52	tangent(x)	tan(x)	Tangent trigonometric function
53	arcsine(x)	asin(x)	Inverse sine function
54	arccosine(x)	acos(x)	Inverse cosine function
55	arctangent(x)	atan(x)	Inverse tangent function