KHULNA UNIVERSITY OF ENGINEERING & TECHNOLOGY

Department of Computer Science and Engineering

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:

Tawhidul Hasan

Roll No: 2107004

Year: Third

Semester: Second

Department of Computer Science and Engineering (CSE)

Khulna University of Engineering & Technology (KUET)

Date of Submission: January 12, 2026

Objectives:

- To design a custom high-level programming language using a wizard/magic theme
- To implement a lexical analyzer using Flex for token generation
- To implement a syntax analyzer using Bison for grammar parsing
- To map wizard-themed keywords and symbols **to** equivalent C language constructs

Introduction:

Compiler design is a fundamental area of computer science that focuses on translating high-level programming languages into machine-understandable code. This project proposes the design and implementation of a Wizard-Themed Programming Language, inspired by magical spellcasting concepts, using Flex for lexical analysis and Bison for syntax analysis.

The proposed language aims to make programming more engaging, readable, and creative by replacing traditional programming keywords with wizard-inspired terminology such as spells, potions, incantations, and enchantments. Internally, the language constructs will be mapped to equivalent C language constructs, enabling execution through semantic actions.

Flex (Fast Lexical Analyzer)

Flex is a tool used to generate a lexical analyzer. It reads the source program and converts character streams into meaningful tokens such as keywords, identifiers, operators, and literals. In this project, Flex will recognize wizard-themed keywords and symbols.

Bison (Parser Generator)

Bison is a parser generator used to analyze the syntax structure of the program. It checks whether the token sequence produced by Flex follows the defined grammar rules. Bison will be used to validate control structures, expressions, and function definitions of the wizard language.

SL NO	***Keyword/Symbol***	***String/Keyword / Symbol in C***	***Description***
1.	#summon	#include	Include header files
2.	#engrave	#define	Macro Definition
3.	RuneInt	int	Integer data type
4.	RuneFloat	float	Float data type
5.	RuneDouble	double	Double data type
6.	RuneLong	long integer	Long Integer data type
7.	RuneChar	char	Character type data
8.	RuneBool	bool	Bool type data
9.	Voidum	void	no return type
10.	break_spell	break	Break statement
11.	Lumos()	printf()	Display output (bring light)
12.	Murmurus()	scanf()	Take input (whisper input)
13.	%%	//	Single line comment
14.	%* … … … *%	/* … … …*/	Multi line comment

15.	Spell add(type a, type b) :->
	{
	}
	EndSpell
		returnType functionName(type1 argument1, type2 argument2, ...);	Function declaration

16.	if_charm(expr)	if (test expression)	If statement
17.	else_charm(expr)	else if (test expression)	Else if statement

18.	otherwise	else	Else statement
19.	TimeTurner(init:cond:update)	for (initialize; condition; update) {}	For loop
20.	while_spell(expr)	while (testExpression){}	While loop
21.	DoFirst { } … while_spell(cond)	do { }while(testExpression)	Do while loop
22.	keep_casting	continue	continue statement
23.	SortingHat	switch	switch statement
24.	House	case	Case Label
25.	Muggle	default	default case
26.	.	;	end statement
27.	pluso	+	addition operator
28.	Minuo	-	subtraction operator
29.	Multo	*	multiplication operator
30.	Divio	/	division operator
31.	Modio	%	modulus operator
32.	AND	&&	logical AND operator
33.	OR	||	logical OR operator
34.	NOT	!	logical NOT operator
35.	XOR	^	logical XOR operator
36.	powo	pow()	exponentiation operator
37.	Radix()	sqrt()	Square root function
38.	Floorus()	floor()	Floor function
39.	Ceilus()	ceil()	Ceil function

40.	Absolutus()	abs()	Absolute function
41.	Logus()	log()	Logarithm function
42.	Sinus()	sin()	Sine function
43.	Cosinus()	cos()	Cosine function
44.	Tanus()	tan()	Tangent function
45.	=	=	Assignment operator
46.	<	<	Less then
47.	>	>	Greater than
48.	<=	<=	Greater or Equal
49.	>=	>=	Less or Equal
50.	Equals	==	Equal
51.	NotEquals	!=	Not equal
52.	:->		return type for function
53.	ExpectoReturnum	return	return statement
54.	asine()	asin()	the arc sine (inverse sine) of a number in radians
55.	acosine()	acos()	the arc cosine (inverse cosine) of a number in
56.	atan()	atan()	the arc tangent (inverse tangent) of a number in radians
57.	IDENTIFIER	[A-Za-z_]+[A-Za-z_0-9]*	Variable declaration
58.	DIGIT	[0-9]	Digits
59.	FLOATING	[0-9]+[.][0-9]+	Floating number
60.	{	{	opening parenthesis

61.	}	}	closing parenthesis
62.	StartoMagica()	main()	main function
63.	up	++	increment
64.	down	--	decrement