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:

Sarwad Hasan Siddiqui

Roll No: 2107006

Year: Third

Semester: Second

Department of Computer Science and Engineering (CSE)

Khulna University of Engineering & Technology (KUET)

Date of Submission: January 12, 2026

ASTROSCRIPT: A Language for Space Missions

Brief Description:

AstroScript is a **mission-oriented** programming language where each program is structured as a **space mission**, beginning with a launch and ending with either successful completion or abort. The language redefines common programming constructs such as variables, functions, loops, conditionals, and data structures using intuitive, space-inspired keywords while maintaining clear semantics and static typing. **AstroScript** is designed to be implemented using **Flex** and **Bison** and supports both primitive and complex data types, including structured modules, collections, and enumerated mission states. In addition to the language implementation, the project includes a **visual compiler simulator** that illustrates the complete compilation and execution process, enabling users to observe tokenization, parsing, intermediate representation, and step-by-step execution in an interactive manner.

Keywords Mapping:

SL NO	Keyword / Symbol (AstroScript)	String / Keyword / Symbol in C / C++	Description
1	mission	main()	Entry point of an AstroScript program
2	launch	start of main	Begins mission execution
3	success	return 0;	Successful mission termination
4	abort	exit(1)	Abnormal mission termination
5	count	int	Integer type for discrete values
6	real	float	Floating-point type
7	precise	double	High-precision numeric type
8	flag	bool	Boolean signal type
9	symbol	char	Single character type
10	voidspace	void	No return type
11	telemetry <type> <id>	variable	Mutable variable / class member
12	limit <type> <id>	const	Immutable constant / const member
13	:=	=	Assignment operator
14	.	;	End of statement
15	broadcast()	printf()	Output to console
16	receive()	scanf()	Input from user
17	alarm()	printf()	Critical warning output

SL NO	**Keyword / Symbol (AstroScript)**	**String / Keyword / Symbol in C / C++**	**Description**
18	$$	//	Single-line comment
19	$* ... *$	/* ... */	Multi-line comment
20	command name(...) :-> type	method / function	Defines a function or class method
21	back	return	Return statement
22	verify (condition)	if	Conditional check
23	else_verify (condition)	else if	Alternate condition
24	otherwise	else	Default condition
25	orbit while (condition)	while	Loop with condition
26	orbit times (i:c:u)	for	Fixed iteration loop
27	stage_sep	break	Exit loop
28	coast	continue	Skip loop iteration
29	scenario (expression)	switch	Multi-path decision
30	trajectory	case	Switch case
31	fallback	default	Default case
32	add	+	Addition
33	minus	-	Subtraction
34	mul	*	Multiplication
35	divide	/	Division
36	mod	%	Modulus
37	**	pow()	Exponentiation
38	AND	&&	Logical AND
39	OR	`	
40	NOT	!	Logical NOT
41	XOR	^	Logical XOR
42	<	<	Less than
43	>	>	Greater than
44	<=	<=	Less than or equal
45	>=	>=	Greater than or equal
46	==	==	Equality check
47	!=	!=	Not equal
48	root()	sqrt()	Square root
49	flr()	floor()	Floor
50	ceil()	ceil()	Ceiling
51	abs()	abs()	Absolute value
52	logarithm()	log()	Logarithm
53	sine()	sin()	Sine
54	cosine()	cos()	Cosine

SL NO	Keyword / Symbol (AstroScript)	String / Keyword / Symbol in C / C++	Description
55	tan()	tan()	Tangent
56	asine()	asin()	Inverse sine
57	acosine()	acos()	Inverse cosine
58	atan()	atan()	Inverse tangent
59	prime()	user-defined	Prime number check
60	wait <n> tick	delay	Time-based execution pause
61	module	class / struct	Defines a class
62	deploy	object creation	Instantiate an object
63	extends	inheritance	Single inheritance
64	public	public	Public access specifier
65	private	private	Private access specifier
66	this	this	Reference to current object
67	fleet	array	Collection of objects or values
68	mode	enum	Enumerated type
69	alias	typedef	Type alias
70	{	{	Block start
71	}	}	Block end