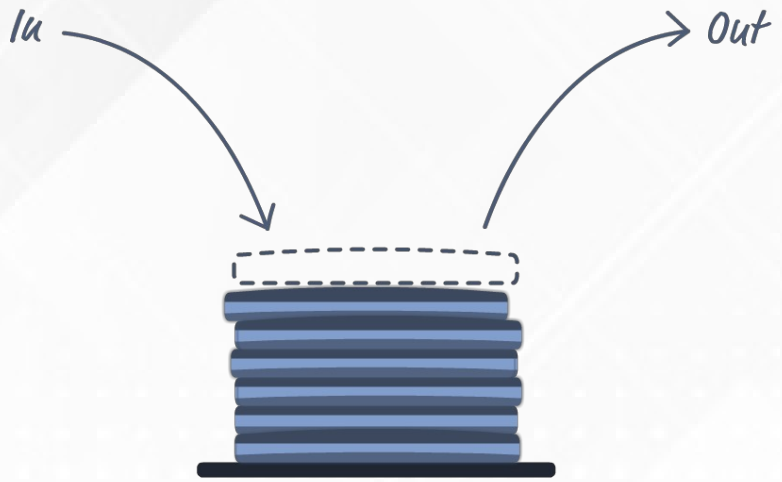
Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Unit-1 (Part 3)

Stack

Linear Data Structure



Prof. Shruti Maniar
Computer Engineering Department
Darshan University, Rajkot

shruti.maniar@darshan.ac.in
+91 97277 47317 (CE Department)

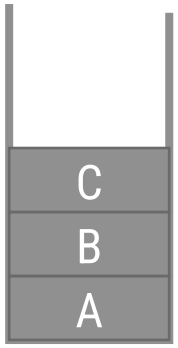
Topics to be covered

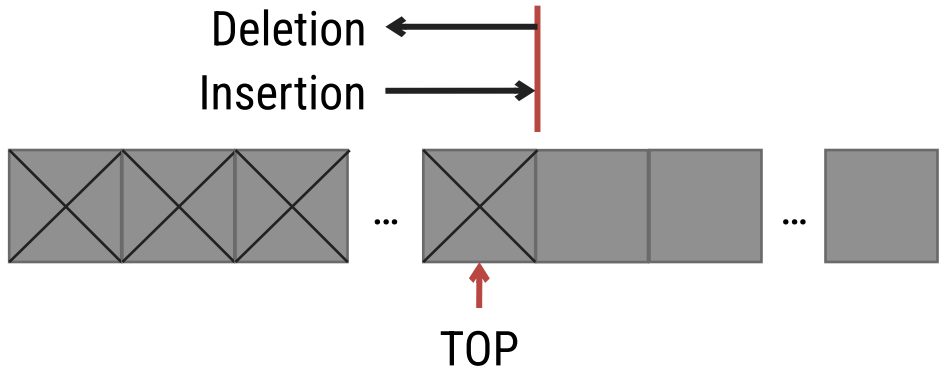
- Stack – Introduction
- Operations on Stack
- Application of Stack
  - Check the given grammar for the input string
  - Polish Notations and their compilation
    - Infix to Postfix Conversion
    - Infix to Prefix Conversion
    - Evaluation of Postfix Expression
    - Evaluation of Prefix Expression
    - Recursion (Factorial)

Stack

Stack

- A linear list which allows insertion and deletion of an element at **one end only** is called stack.
- The **insertion operation** is called as PUSH and **deletion operation** as POP.
- The **most accessible elements** in stack is known as top.
- The elements can only be removed in the opposite orders from that in which they were added to the stack.
- Such a linear list is referred to as a LIFO (Last In First Out) list.





Stack

- A pointer TOP keeps track of the top element in the stack.
- Initially, when the **stack is empty**, TOP has a value of "zero".
- Each time a **new element is inserted** in the stack, the pointer is incremented by "one" before, the element is placed on the stack.
- The pointer is decremented by "one" each time a **deletion is made** from the stack.

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Operations on Stack

Push – Pop – Peep - Change

Procedure : PUSH (S, TOP, X)

- This procedure inserts an element **X** to the top of a stack.
- Stack is represented by a vector **S** containing **N** elements.
- A pointer **TOP** represents the top element in the stack.

```
1. [Check for stack overflow]
      If    TOP ≥ N
      Then  write ('STACK OVERFLOW')
            Return
2. [Increment TOP]
      TOP ← TOP + 1
3. [Insert Element]
      S[TOP] ← X
4. [Finished]
      Return
```

Stack is empty, TOP = 0, N=3

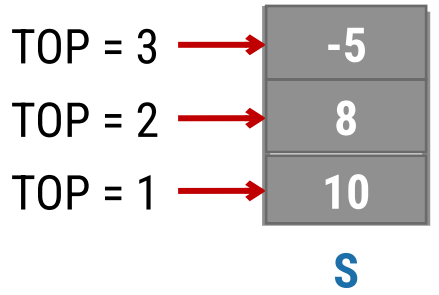
PUSH(S, TOP, 10)

PUSH(S, TOP, 8)

PUSH(S, TOP, -5)

PUSH(S, TOP, 6)

Overflow



Function : POP (S, TOP)

- This function ***removes & returns*** the top element from a stack.
- Stack is represented by a vector **S** containing **N** elements.
- A pointer **TOP** represents the top element in the stack.

```
1. [Check for stack underflow]
      If     TOP = 0
      Then   write ('STACK UNDERFLOW')
             Return (0)
2. [Decrement TOP]
      TOP ← TOP - 1
3. [Return former top element of
   stack]
      Return(S[TOP + 1])
```

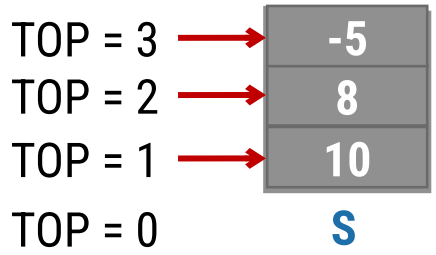
POP(S, TOP)

POP(S, TOP)

POP(S, TOP)

POP(S, TOP)

Underflow



Darshan UNIVERSITY

Function : PEEP (S, TOP, I)

- This function returns the value of the **Ith** element from the **TOP** of the stack. The element is not deleted by this function.
- Stack is represented by a vector **S** containing **N** elements.

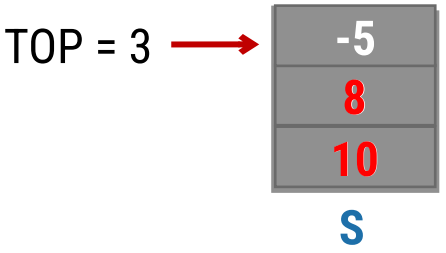
```
1. [Check for stack underflow]
      If    TOP-I+1 ≤ 0
      Then  write ('STACK UNDERFLOW')
            Return (0)
2. [Return Ith element from top
    of the stack]
      Return(S[TOP-I+1])
```

PEEP (S, TOP, 2)

PEEP (S, TOP, 3)

PEEP (S, TOP, 4)

Underflow



PROCEDURE : CHANGE (S, TOP, X, I)

- This procedure changes the value of the **Ith** element from the top of the stack to **X**.
- Stack is represented by a vector **S** containing **N** elements.

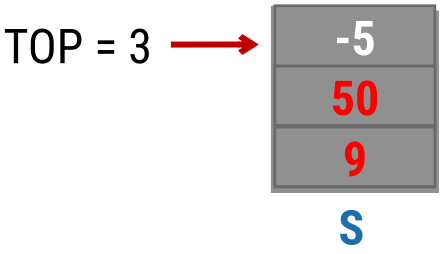
```
1. [Check for stack underflow]
      If    TOP-I+1 ≤ 0
      Then  write ('STACK UNDERFLOW')
            Return
2. [Change Ith element from top
    of the stack]
      S[TOP-I+1] ← X
3. [Finished]
      Return
```

CHANGE (S, TOP, 50, 2)

CHANGE (S, TOP, 9, 3)

CHANGE (S, TOP, 25, 8)

Underflow



Programming of Stack using an Array

```
#include<stdio.h>
int top=-1;
int MAX=3;
int  S[MAX] ;
void push(int x)
{
                                Array starts from index : 0
        if(top>=MAX-1)
        {
                printf("Stack Overflow");
                return;
        }
        top++;
        S[top]=x;
}
void main(){
        push(10); push(20); push(30); push(40);
}
```

```
Push(S, TOP, x)

1. [Check for stack overflow]
      If    TOP ≥ N   Array started from index :1
      Then  write ('STACK OVERFLOW')
            Return
2. [Increment TOP]
      TOP ← TOP + 1
3. [Insert Element]
      S[TOP] ← X
4. [Finished]
      Return
```

Check the grammar of the input string

Stack Application

Algorithm: RECOGNIZE

Write an algorithm which will check that the given string belongs to following grammar or not.

$L = \{wcw^R \mid w \in \{a,b\}^*\}$ (Where w^R is the reverse of w)

Example of valid strings : abcba aabbcbbaa

Example of Invalid strings: aabcaab

- Given an input string named ***STRING*** on the alphabet ***{a, b, c}*** which contains a blank in its rightmost character position.
- Function **NEXTCHAR** returns the next symbol in STRING.
- This algorithm determines whether the contents of STRING belong to the above language.
- The vector **S** represents the stack and **TOP** is a pointer to the top element of the stack.

Algorithm: RECOGNIZE

1. [Initialize stack by placing a letter ‘c’ on the top]

```
TOP ← 1
S [TOP] ← ‘c’
```

2. [Get and PUSH symbols until either ‘c’ or blank is encountered]

```
NEXT ← NEXTCHAR (STRING)
Repeat while NEXT ≠ ‘c’
      If   NEXT = ‘ ‘
      Then Write (‘Invalid
String’)
           Exit
   Else Call PUSH (S, TOP, NEXT)
           NEXT ← NEXTCHAR (STRING)
```

3. [Scan characters following ‘c’; Compare them to the characters on stack]

```
Repeat While S [TOP] ≠ ‘c’
      NEXT ← NEXTCHAR (STRING)
      X ← POP (S, TOP)
If      NEXT ≠ X
Then Write(‘INVALID STRING’)
        Exit
```

4. [Next symbol must be blank]

```
NEXT ← NEXTCHAR (STRING)
If     NEXT = ‘ ‘
Then Write (‘VALID STRING’)
Else Write (‘INVALID STRING’)
```

5. [Finished]

```
Exit
```

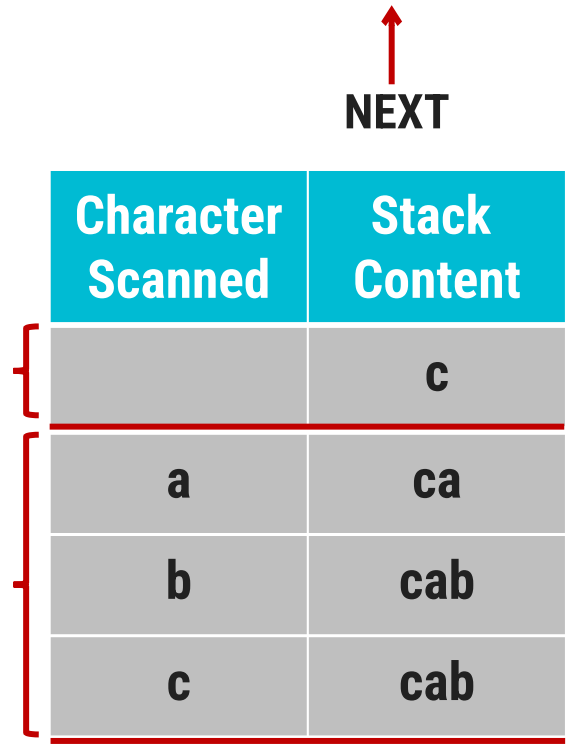
Darshan UNIVERSITY

Algorithm: RECOGNIZE

```
1. [Initialize stack by placing
    a letter 'c' on the top]
   TOP ← 1
   S [TOP] ← 'c'

2. [Get and PUSH symbols until
   either c' or blank is
   encountered]
   NEXT ← NEXTCHAR (STRING)
   Repeat while NEXT ≠ 'c'
         If   NEXT = ' '
         Then Write ('Invalid
   String')
              Exit
         Else Call PUSH (S, TOP,
   NEXT)
              NEXT ← NEXTCHAR
   (STRING)
```

Input String: a b c b a □



Algorithm: RECOGNIZE

```
3. Scan characters following 'c';
   Compare them to the characters on stack
   Repeat While S[TOP] ≠ 'c'
      NEXT ← NEXTCHAR (STRING)
      X ← POP (S, TOP)
      If NEXT ≠ X
      Then Write('Invalid String')
           Exit

4. [Next symbol must be blank]
   NEXT ← NEXTCHAR (STRING)
   If  NEXT = ' '
   Then      Write ('VALID STRING')
   Else      Write ('INVALID STRING')
```

Input String: a b c b a □

NEXT

	Character Scanned	Stack Content
1		c
2	a	ca
	b	cab
	c	cab
3	b	ca
	a	c
	□	

Algorithm : RECOGNIZE

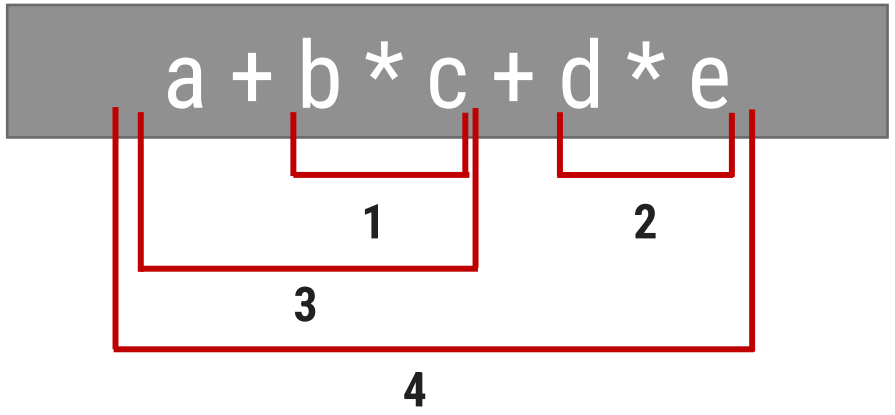
- Write an algorithm to determine if an input character string is of the form a^ib^i ***where i>=1***
- i.e. number of **a** ***should be equal*** to no of **b**

Polish Notations

Stack Application

Polish Expression & their Compilation

- Evaluating Infix Expression



- A **repeated scanning** **from left to right is needed** as operators appears inside the operands.
- ***Repeated scanning is avoided*** if the **infix expression** is first **converted** to an equivalent parenthesis free ***prefix or suffix (postfix) expression***.
- **Prefix Expression: Operator**, Operand, Operand
- **Postfix Expression:** Operand, Operand, **Operator**

Polish Notation

- This type of notation is known **Lukasiewicz Notation** or **Polish Notation** or **Reverse Polish Notation** due to Polish logician *Jan Lukasiewicz*.
- In both **prefix** and **postfix** equivalents of an infix expression, the ***variables are in same relative position***.
- The expressions in postfix or prefix form are ***parenthesis free*** and operators are rearranged according to rules of precedence for operators.

Polish Notation

Sr.	Infix	Postfix	Prefix
1	a	a	a
2	a + b	a b +	+ a b
3	a + b + c	a b + c +	+ + a b c
4	a + (b + c)	a b c + +	+ a + b c
5	a + (b * c)	a b c * +	+a * b c
6	a * (b + c)	a b c + *	* a + b c
7	a * b * c	a b *c*	** a b c

Infix to Postfix Conversion

Stack Application

Finding Rank of any Expression

E = (A + B * C / D - E + F / G / (H + I))

Note: R = *Rank*, Rank of Variable *= 1*, Rank of binary operators *= -1*

Rank (E) = R(A) + R(+) + R(B) + R(*) + R(C) + R (/) + R(D) + R(-) + R(E) + R(+) + R(F) + R(/) + R(G) + R(/) + R(H) + R(+) + R(I)

Rank (E) = 1 + **(-1)** + 1 + **(-1)** + 1 + **(-1)** + 1 + **(-1)** + 1 + **(-1)** + 1 + **(-1)** + 1 + **(-1)** + 1 + **(-1)** + 1

Rank (E) = 1

Any Expression is valid if Rank of that expression is 1

Convert Infix to Postfix Expression

Symbol	Input precedence function (F)	Stack precedence function (G)	Rank function (R)
+, -	1	2	-1
*, /	3	4	-1
^	6	5	-1
Variables	7	8	1
(	9	0	-
)	0	-	-

Algorithm : REVPOL

- Given an input string **INFIX** containing an infix expression which has been padded on the right with **')'**.
- This algorithm ***converts INFIX into reverse polish*** and places the result in the string **POLISH**.
- All symbols have precedence value given by the table.
- Stack is represented by a vector **S, TOP** denotes the top of the stack, algorithm **PUSH** and **POP** are used for stack manipulation.
- Function **NEXTCHAR** returns the next symbol in given input string.
- The integer variable **RANK** contains the rank of expression.
- The string variable **TEMP** is used for temporary storage purpose.

```
1. [Initialize Stack]
    TOP ← 1
    S[TOP] ← '('
2. [Initialize output string and rank count]
    POLISH ← ''
    RANK ← 0
3. [Get first input symbol]
    NEXT ← NEXTCHAR(INFIX)
4. [Translate the infix expression]
    Repeat thru step 7
        while NEXT != ' '
```

Symbol	IPF (F)	SPF (G)	RF (R)
+, -	1	2	-1
*, /	3	4	-1
^	6	5	-1
Variables	7	8	1
(	9	0	-
)	0	-	-

```
5. [Remove symbols with greater precedence from stack]
    IF   TOP < 1
    Then write ('INVALID')
            EXIT
    Repeat while G(S[TOP]) > F(NEXT)
            TEMP ← POP (S, TOP)
            POLISH ← POLISH O TEMP
            RANK ← RANK + R(TEMP)
            IF   RANK <1
            Then write ('INVALID')
                 EXIT
6. [Are there matching parentheses]
    IF   G(S[TOP]) != F(NEXT)
    Then call PUSH (S,TOP, NEXT)
    Else POP (S,TOP)
7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
8. [Is the expression valid]
    IF   TOP != 0 OR RANK != 1
    Then write ('INVALID')
    Else write ('VALID')
```

(a+b^c^d)*(e+f/d))

NEXT

1. [Initialize Stack]
   TOP ← 1
   S[TOP] ← ‘(’
2. [Initialize output string and rank count]
   POLISH ← ‘’
   RANK ← 0
3. [Get first input symbol]
   NEXT ← NEXTCHAR(INFIX)

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(			

(a + b ^ c ^ d) * (e + f / d))

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
   stack]
   IF    TOP < 1
   Then write ('INVALID')
        EXIT
   Repeat while G(S[TOP]) > F(NEXT)
       TEMP ← POP (S, TOP)
       POLISH ← POLISH O TEMP
       RANK ← RANK + R(TEMP)
       IF    RANK <1
       Then write ('INVALID')
            EXIT

6. [Are there matching parentheses]
    IF    G(S[TOP]) != F(NEXT)
    Then call PUSH (S,TOP, NEXT)
    Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1

```
( a + b ^ c ^ d ) * ( e + f / d ) )
```

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
    stack]
    IF    TOP < 1
    Then write ('INVALID')
         EXIT
    Repeat while G(S[TOP]) > F(NEXT)
        TEMP ← POP (S, TOP)
        POLISH ← POLISH O TEMP
        RANK ← RANK + R(TEMP)
        IF    RANK <1
        Then write ('INVALID')
             EXIT

6. [Are there matching parentheses]
     IF    G(S[TOP]) != F(NEXT)
     Then call PUSH (S,TOP, NEXT)
     Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	(	abcd^^+	1
*	(*	abcd^^+	1

(a+b^c^d)*(e+f/d))

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
    stack]
    IF    TOP < 1
    Then write ('INVALID')
         EXIT
    Repeat while G(S[TOP]) > F(NEXT)
        TEMP ← POP (S, TOP)
        POLISH ← POLISH O TEMP
        RANK ← RANK + R(TEMP)
        IF    RANK <1
        Then write ('INVALID')
             EXIT

6. [Are there matching parentheses]
     IF    G(S[TOP]) != F(NEXT)
     Then call PUSH (S,TOP, NEXT)
     Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1
*	(*	abcd^^+	1
(	(*(	abcd^^+	1

```
(a+b^c^d)*(e+f/d))
```

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
    stack]
    IF    TOP < 1
    Then write ('INVALID')
         EXIT
    Repeat while G(S[TOP]) > F(NEXT)
        TEMP ← POP (S, TOP)
        POLISH ← POLISH O TEMP
        RANK ← RANK + R(TEMP)
        IF    RANK <1
        Then write ('INVALID')
             EXIT

6. [Are there matching parentheses]
     IF    G(S[TOP]) != F(NEXT)
     Then call PUSH (S,TOP, NEXT)
     Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1
*	(*	abcd^^+	1
(	(*(	abcd^^+	1
e	(*(e	abcd^^+	1

```
( a + b ^ c ^ d ) * ( e + f / d ) )
```

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ‘ ‘

5. [Remove symbols with greater precedence from
    stack]
    IF    TOP < 1
    Then write (‘INVALID’)
         EXIT
    Repeat while G(S[TOP]) > F(NEXT)
        TEMP ← POP (S, TOP)
        POLISH ← POLISH O TEMP
        RANK ← RANK + R(TEMP)
        IF     RANK <1
        Then write (‘INVALID’)
             EXIT

6. [Are there matching parentheses]
     IF    G(S[TOP]) != F(NEXT)
     Then call PUSH (S,TOP, NEXT)
     Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1
*	(*	abcd^^+	1
(	(*(	abcd^^+	1
e	(*(e	abcd^^+	1
+	(*(+	abcd^^+e	2

```
(a+b^c^d)*(e+f/d))
```

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
    stack]
    IF    TOP < 1
    Then write ('INVALID')
         EXIT
    Repeat while G(S[TOP]) > F(NEXT)
        TEMP ← POP (S, TOP)
        POLISH ← POLISH O TEMP
        RANK ← RANK + R(TEMP)
        IF    RANK <1
        Then write ('INVALID')
             EXIT

6. [Are there matching parentheses]
     IF    G(S[TOP]) != F(NEXT)
     Then call PUSH (S,TOP, NEXT)
     Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1
*	(*	abcd^^+	1
(	(*(	abcd^^+	1
e	(*(e	abcd^^+	1
+	(*(+	abcd^^+e	2
f	(*(+f	abcd^^+e	2

```
(a+b^c^d)*(e+f/d))
```

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
    stack]
    IF    TOP < 1
    Then write ('INVALID')
         EXIT
    Repeat while G(S[TOP]) > F(NEXT)
        TEMP ← POP (S, TOP)
        POLISH ← POLISH O TEMP
        RANK ← RANK + R(TEMP)
        IF    RANK <1
        Then write ('INVALID')
             EXIT

6. [Are there matching parentheses]
     IF    G(S[TOP]) != F(NEXT)
     Then call PUSH (S,TOP, NEXT)
     Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1
*	(*	abcd^^+	1
(	(*(	abcd^^+	1
e	(*(e	abcd^^+	1
+	(*(+	abcd^^+e	2
f	(*(+f	abcd^^+e	2
/	(*(+/	abcd^^+ef	3

```
(a+b^c^d)*(e+f/d))
```

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
    stack]
    IF    TOP < 1
    Then write ('INVALID')
         EXIT
    Repeat while G(S[TOP]) > F(NEXT)
        TEMP ← POP (S, TOP)
        POLISH ← POLISH O TEMP
        RANK ← RANK + R(TEMP)
        IF    RANK <1
        Then write ('INVALID')
             EXIT

6. [Are there matching parentheses]
     IF    G(S[TOP]) != F(NEXT)
     Then call PUSH (S,TOP, NEXT)
     Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1
*	(*	abcd^^+	1
(	(*(	abcd^^+	1
e	(*(e	abcd^^+	1
+	(*(+	abcd^^+e	2
f	(*(+f	abcd^^+e	2
/	(*(+/	abcd^^+ef	3
d	(*(+/d	abcd^^+ef	3

(a+b^c^d)*(e+f/d))

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
   stack]
   IF    TOP < 1
   Then write ('INVALID')
        EXIT
   Repeat while G(S[TOP]) > F(NEXT)
       TEMP ← POP (S, TOP)
       POLISH ← POLISH O TEMP
       RANK ← RANK + R(TEMP)
       IF    RANK <1
       Then write ('INVALID')
            EXIT

6. [Are there matching parentheses]
    IF    G(S[TOP]) != F(NEXT)
    Then call PUSH (S,TOP, NEXT)
    Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1
*	(*	abcd^^+	1
(	(*(	abcd^^+	1
e	(*(e	abcd^^+	1
+	(*(+	abcd^^+e	2
f	(*(+f	abcd^^+e	2
/	(*(+/	abcd^^+ef	3
d	(*(+/d	abcd^^+ef	3
)	(*(+/	abcd^^+efd/+	3

(a+b^c^d)*(e+f/d))

NEXT

```
4. [Translate the infix expression]
    Repeat thru step 7
          while NEXT!= ' '

5. [Remove symbols with greater precedence from
   stack]
   IF    TOP < 1
   Then write ('INVALID')
        EXIT
   Repeat while G(S[TOP]) > F(NEXT)
       TEMP ← POP (S, TOP)
       POLISH ← POLISH O TEMP
       RANK ← RANK + R(TEMP)
       IF    RANK <1
       Then write ('INVALID')
            EXIT

6. [Are there matching parentheses]
    IF    G(S[TOP]) != F(NEXT)
    Then call PUSH (S,TOP, NEXT)
    Else POP (S,TOP)

7. [Get next symbol]
    NEXT ← NEXTCHAR(INFIX)
```

Input Symbol	Content of stack	Reverse polish expression	Rank
	(		0
(	((		0
a	((a		0
+	((+	a	1
b	((+b	a	1
^	((+^	ab	2
c	((+^c	ab	2
^	((+^^	abc	3
d	((+^^d	abc	3
)	((	abcd^^+	1
*	(*	abcd^^+	1
(	(*(	abcd^^+	1
e	(*(e	abcd^^+	1
+	(*(+	abcd^^+e	2
f	(*(+f	abcd^^+e	2
/	(*(+/	abcd^^+ef	3
d	(*(+/d	abcd^^+ef	3
)	(*	abcd^^+efd/+	2
)	(	abcd^^+efd/+*	1

General Infix to Postfix Conversion

Add ‘)’ to the end of the infix expression.

Push ‘(‘ on to the stack.

Repeat until each character in the infix notation is scanned

Precedence
^, $
*, / ,%
-, +

1. If an **operand** (whether a number or a character) is encountered, add it to the postfix expression
2. If a left parenthesis ‘(’, is encountered, push it on the stack
3. If a **right parenthesis ‘)’**, is encountered, then
   - Repeatedly **pop** from stack and add it to the postfix expression until a **‘(‘** is encountered.
   - **Discard** the ‘ (‘.
4. If an **operator O** , is encountered, then
   - **Repeatedly pop** from the stack and add each operator (popped from the stack) to the postfix expression which has the **same or a higher precedence than O**.(**except ^ :** Only higher precedence operators)
   - **Push the operator O** to the stack.

Darshan UNIVERSITY

योगः कर्मसु कौशलम्

Example : Infix to Postfix : a + b * c – d / e

a + b * c – d / e)

Character Scanned	Stack	Postfix Expression
	(	Empty
a	(	a
+	(+	a
b	(+	a b
*	(+ *	a b
c	(+ *	a b c
-	(-	a b c * +
d	(-	a b c * + d
/	(- /	a b c * + d
e	(- /	a b c * + d
)	Empty	a b c * + d e / -

Precedence
^
*, /
-, +

Exercise : Infix to Postfix & Prefix Conversion

Sr.	Infix Expressions
1	a + b * c – d / e * h
2	A ^ B – C * D + E ^ F / G
3	A + B – C * D * E ^ F ^ G
4	2 * 3 / (2 – 1) + 5 * 3
5	A + (B * (C – D) / E)
6	(A + B) * C + D / (B + A * C) + D
7	((a+b^c^d)*(e+f/d))
8	(A + B * C / D - E + F / G / (H + I))
9	a + ((b * c) / (d – e))
10	A + (B * C – (D / E – F) * G) * H

Exercise : Infix to Postfix

Sr.	Infix Expressions	Postfix Expression
1	a + b * c – d / e * h	a b c * + d e / h * -
2	A ^ B – C * D + E ^ F / G	A B ^ C D * - E F ^ G / +
3	A + B – C * D * E ^ F ^ G	A B + C D * E F G ^ ^ * -
4	2 * 3 / (2 – 1) + 5 * 3	2 3 * 2 1 - / 5 3 * +
5	A + (B * (C – D) / E)	A B C D - * E / +
6	(A + B) * C + D / (B + A * C) + D	A B + C * D B A C * + / + D +
7	((a+b^c^d)*(e+f/d))	a b c d ^ ^ + e f d / + *
8	(A + B * C / D - E + F / G / (H + I))	A B C * D / + E - F G / H I + / +
9	a + ((b * c) / (d – e))	a b c * d e - / +
10	A + (B * C – (D / E – F) * G) * H	A B C * D E / F – G * - H * +

General Infix to Prefix Conversion

1. Reverse the infix expression.
   Note : while reversing **each '(' will become ')'** and each **')' becomes '('**.

2. Convert the reversed infix expression to **"nearly" postfix** expression.
   - While converting, **instead of popping operators with greater than or equal precedence**, here we will **only pop the operators from stack that have greater precedence**. (**except ^ : same or higher** precedence operators)
   - Push the operator

3. Reverse the postfix expression.

Convert Infix to Prefix Expression

Symbol	Input precedence function (F)	Stack precedence function (G)	Rank function (R)
+, -	**2**	**1**	-1
***, /**	**4**	**3**	-1
^	**5**	**6**	-1
Variables	7	8	1
(	9	0	-
)	0	-	-

Exercise : Infix to Prefix

Sr.	Infix Expressions	Prefix Expressions
1	a + b * c – d / e * h	- + a * b c * / d e h
2	A ^ B – C * D + E ^ F / G	+ - ^ A B * C D / ^ E F G
3	A + B – C * D * E ^ F ^ G	- + A B * * C D ^ E ^ F G
4	2 * 3 / (2 – 1) + 5 * 3	+ / * 2 3 – 2 1 * 5 3
5	A + (B * (C – D) / E)	+ A / * B – C D E
6	(A + B) * C + D / (B + A * C) + D	+ + * + A B C / D + B * A C D
7	((a+b^c^d)*(e+f/d))	* + a ^ b ^ c d + e / f d
8	(A + B * C / D - E + F / G / (H + I))	+ - + A / * B C D E / / F G + H I
9	a + ((b * c) / (d – e))	+ a / * b c – d e
10	A + (B * C – (D / E – F) * G) * H	+ A * - * B C * - / D E F G H

Evaluation of Postfix Expression

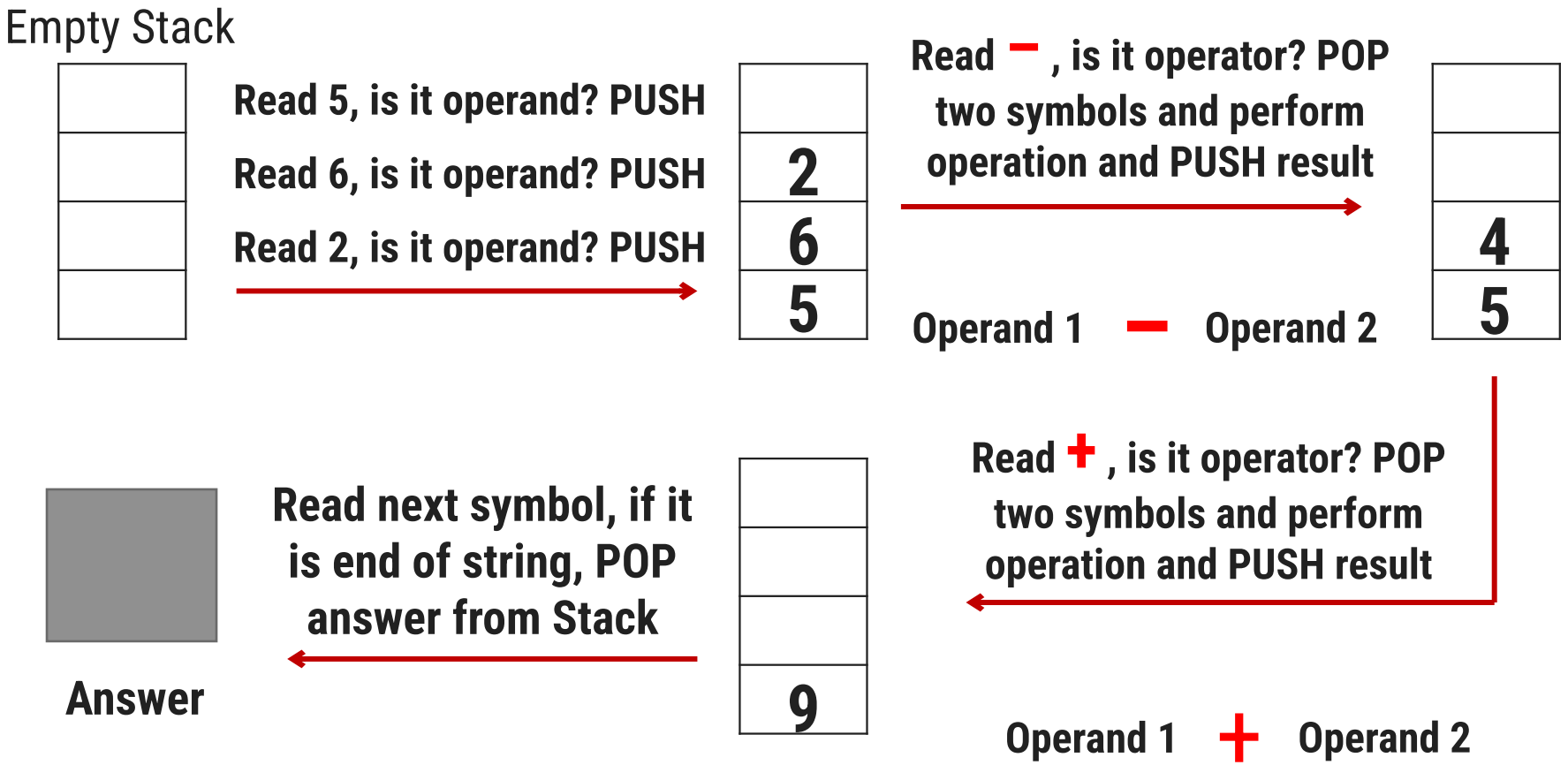
Stack Application

Evaluation of Postfix Expression

- Each **operator** in **postfix** string **refers** to the *previous two operands* in the string.
- Each time we **read** an **operand**, we **PUSH** it onto **Stack**.
- When we reach an **operator**, its **operands** will be **top two elements** on the stack.
- We can then **POP** these two elements, perform the indicated operation on them and PUSH the result on the stack so that it will available for use as an operand for the next operator.

Evaluation of Postfix Expression

Evaluate Expression: 5 6 2 - +



Evaluation of Postfix Expression : 5,4,6,+,*,4,9,3,/,+,*

Character Scanned	Stack	Operand 1	Operand 2	Value
5	5			
4	5, 4			
6	5, 4, 6			
+	5, 10	4	6	10
*	50	5	10	50
4	50, 4			
9	50, 4, 9			
3	50, 4, 9, 3			
/	50, 4, 3	9	3	3
+	50, 7	4	3	7
*	350	50	7	350

Answer = pop() = 350

Algorithm: EVALUATE_POSTFIX

- Given an input string **POSTFIX** representing postfix expression.
- This algorithm evaluates postfix expression and put the result into variable **VALUE**.
- Stack is represented by a vector **S, TOP** denotes the top of the stack, Algorithm **PUSH** and **POP** are used for stack manipulation.
- Function **NEXTCHAR** returns the next symbol in given input string.
- **OPERAND1**, **OPERAND2** and **TEMP** are used for temporary variables
- **PERFORM_OPERATION** is a function which performs required operation on **OPERAND1** & **OPERAND2**.

Algorithm: EVALUATE_POSTFIX

```
1. [Initialize Stack]
    TOP ← 0
    VALUE ← 0

2. [Evaluate the postfix expression]
   Repeat until last character
       TEMP ← NEXTCHAR (POSTFIX)
       If TEMP is DIGIT
       Then PUSH (S, TOP, TEMP)
       Else OPERAND2 ← POP (S, TOP)
            OPERAND1 ← POP (S, TOP)
            VALUE ← PERFORM_OPERATION(OPERAND1, OPERAND2, TEMP)
            PUSH (S, TOP, VALUE)

3. [Return answer from stack]
   Return (POP (S, TOP))
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Exercise : Evaluation of Postfix Expression

Sr.	Postfix Expressions
1	7 5 2 + * 4 1 1 + / -
2	12, 7, 3, -, /, 2, 1, 5, +, *, +
3	2, 3, 1, *, +, 9, -
4	5, 6, 2, +, *, 12, 4, /, -
5	12, 2, /, 34, 20, -, +, 5, +

Evaluation of Prefix Expression

Stack Application

Algorithm: EVALUATE_PREFIX

- Given an input string **PREFIX** representing prefix expression.
- This algorithm evaluates prefix expression and put the result into variable **VALUE**.
- Stack is represented by a vector **S, TOP** denotes the top of the stack, Algorithm **PUSH** and **POP** are used for stack manipulation.
- Function **NEXTCHAR** returns the next symbol in given input string.
- **OPERAND1**, **OPERAND2** and **TEMP** are used for temporary variables
- **PERFORM_OPERATION** is a function which performs required operation on **OPERAND1** & **OPERAND2**.

Algorithm: EVALUATE_PREFIX

```
1. [Initialize Stack]
    TOP ← 0
    VALUE ← 0

2. [Evaluate the prefix expression]
   Repeat from last character up to first
       TEMP ← NEXTCHAR (PREFIX)
       If   TEMP is DIGIT
       Then PUSH (S, TOP, TEMP)
       Else OPERAND1 ← POP (S, TOP)
            OPERAND2 ← POP (S, TOP)
            VALUE ← PERFORM_OPERATION(OPERAND1, OPERAND2, TEMP)
            PUSH (S, TOP, VALUE)

3. [Return answer from stack]
   Return (POP (S, TOP))
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Evaluation of Prefix Expression : -, *, +, 4, 3, 2, 5

Character Scanned	Stack	Operand 1	Operand 2	Value
5	5			
2	5, 2			
3	5, 2, 3			
4	5, 2, 3, 4			
+	5, 2, 7	4	3	7
*	5,14	7	2	14
-	9	14	5	9

Answer = pop() = 9

Exercise : Evaluation of Prefix Expression

Sr.	Prefix Expressions
1	+, *, 2, +, /, 14, 2, 5, 1
2	-, *, 6, 3, -, 4, 1
3	+, +, 2, 6, +, -, 13, 2, 4
4	-, /, *, 2, *, 5, +, 3, 6, 5, 2
5	-, *, 3, +, 16, 2, /, 12, 6

Recursion

Stack Application

Recursion

A **procedure** that contains a **procedure call to itself** or a procedure **call to second procedure** which eventually **causes the first procedure to be called** is known as **recursive procedure**.

Two important conditions for any recursive procedure	
1	Each time a procedure calls itself it must be nearer in some sense to a solution.
2	There must be a decision criterion for stopping the process or computation.

Two types of recursion	
Primitive Recursion	**Non-Primitive Recursion**
This is **recursive defined function**. E.g. Factorial function	This is **recursive use of procedure**. E.g. Find GCD of given two numbers

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Recursion - Example

```
int factorial(int n)
{
        if(n==0)
                return 1;
        else
                return n*factorial(n-1);
}
```

```
int gcd(int n1, int n2)
{
        if(n2==0)
                return n1;
        else if(n2>n1)
                return gcd(n2,n1);
        else
                return gcd(n2, n1%n2);
}
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Algorithm to find factorial using recursion

- Given integer number **N**
- This algorithm **computes factorial of N**.
- **Stack S** is used to store an activation record associated with each recursive call.
- **TOP** is a pointer to the top element of stack S.
- Each **activation record contains** the current value of **N** and the current return address **RET_ADDE**.
- **TEMP_REC** is also a record which contains two variables **PARAM** & **ADDRESS**.
- **Initially** return address is set to the **main calling address**. PARAM is set to initial value N.

Algorithm: FACTORIAL

```
1. [Save N and return Address]
      CALL PUSH (S, TOP, TEMP_REC)
2. [Is the base criterion found?]
      If     N=0
      then   FACTORIAL ← 1
             GO TO Step 4
      Else   PARAM ← N-1
             ADDRESS ← Step 3
             GO TO Step 1
3. [Calculate N!]
      FACTORIAL← N * FACTORIAL
4. [Restore previous N and return address]
      TEMP_REC ← POP(S,TOP)
      (i.e. PARAM ← N, ADDRESS ← RET_ADDR)
      GO TO ADDRESS
```

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Trace of Algorithm FACTORIAL, N=2

Level Number	Description	Stack Content
Enter Level 1 (main call)	Step 1: PUSH (S,0,(N=2, main address)) Step 2: N!=0 PARAM ← N-1 (1), ADDR ← Step 3	**2** | **Main Address** **TOP**
Enter Level 2 (first recursive call)	Step 1: PUSH (S,1,(N=1, step 3)) Step 2: N!=0 PARAM ← N-1 (3), ADDR ← Step 3	**2** | **1** **Main Address** | **Step 3** **TOP**
Enter Level 3 (second recursive call)	Step 1: PUSH (S,2,(N=0, step 3)) Step 2: N=0 FACTORIAL ← 1 Step 4: POP(A,3) GO TO Step 3	**2** | **1** | **0** **Main Address** | **Step 3** | **Step 3** **TOP** **2** | **1** **Main Address** | **Step 3** **TOP**

Trace of Algorithm FACTORIAL, N=2

Level Number	Description	Stack Content
Return to Level 2	Step 3: FACTORIAL ← 1*1 Step 4: POP (A,2) GO TO Step 3	**2** | **Main Address** **TOP**
Return to Level 1	Step 3: FACTORIAL ← 2*1 Step 4: POP (A,1) GO TO Main Address **TOP = 0**	

Frequently asked questions:

- State algorithms for following operations on Stack: PUSH, POP, PEEP, CHANGE.
- Write an algorithm to evaluate POSTFIX Expression.
- Write an algorithm to convert an infix expression to postfix expression.
- Consider the stack S of characters, where S is allocated 8 memory cells.
  - S: A,C,D, F, K, _, _, _. Sketch the stack after each of the following operations.
  - Pop(), Pop() ,Push(L), Push(P), Pop(), Push(R), Push (S), Pop().
- Solve the Postfix Expression 6 2 3 + - 3 8 2 / + * 2 $ 3 + using Stack.

Darshan UNIVERSITY
योगः कर्मसु कौशलम्

Thank You

Prof. Shruti Maniar
Computer Engineering Department
Darshan University, Rajkot

shruti.maniar@darshan.ac.in
+91 97277 47317 (CE Department)

MARUTI SUZUKI ARENA

MARUTI SUZUKI ARENA

FOR THE DRIVE INSIDE YOU THAT WANTS IT ALL

Maybe it's the way it stands. Unshaken. Self-assured.
Maybe it's the way it moves. As if it owns the road.
The Victoris is for the ones who know when to hold back, and when to go all out.
Who seek out new adventures, new experiences.
Who refuse to settle for anything less than everything.

For the ones who have it all. An SUV that's:

GOT IT ALL

THE CHISELED LOOK THAT SAYS IT ALL

The Victoris speaks in a design language that's impossible to ignore. Its sweeping silhouette and sculpted lines create a stance that's dynamic when in motion and commanding when still.
The seamless tail light stretches across the rear, leaving a signature that lingers long after it's gone.

CRAFTED FOR ALL YOUR SENSES

A cabin tailored for those who know what they want. Everything. With a touch of class.

Its expansive design with dual-tone interiors, panoramic sunroof, vibrant 64-colour ambient lighting, right down to the elegant three-level dashboard with precision stitching; creates a space that serenades all your senses.


64-COLOUR AMBIENT LIGHTING


PANORAMIC SUNROOF


PLUSH LEATHERETTE SEATS

ALL YOU NEED FOR COMPLETE PEACE OF MIND

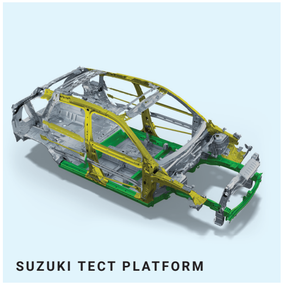
Safety is more than just a feature. It's a promise. It's the quiet assurance that's inherent to every aspect of the Victoris.
Where advanced driver-assist technology works seamlessly with intuitive features to create a drive that feels unshakably secure.
So, while the road may throw its share of surprises, you stay confidently in control.

6 AIRBAGS STANDARD

FRONT PARKING ASSIST SENSOR WITH AUTO 360° CAMERA ACTIVATION

360° VIEW HD CAMERA WITH 11 VIEWS

ELECTRONIC PARKING BRAKE WITH BRAKE HOLD



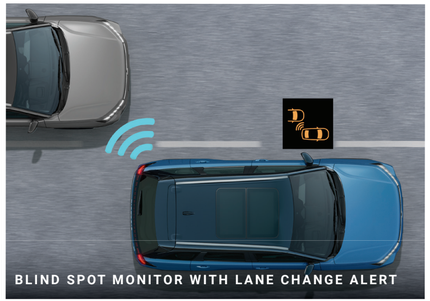
SUZUKI TECT PLATFORM

ADVANCED LEVEL 2 ADAS

Real confidence comes from knowing that your car is always thinking two steps ahead.
The Level 2 ADAS on the Victoris preempts, and protects you, as your ever-present guardian.

LEVEL 2 ADAS* FEATURES

- Lane Keep Assist
- Automatic Emergency Braking
- Forward Collision Warning
- Blind Spot Monitor with Lane Change Alert
- Adaptive Cruise Control with Curve Speed Reduction
- High Beam Assist
- Lane Departure Prevention
- Lane Departure Warning
- Rear Cross Traffic Alert
- Vehicle Sway Warning



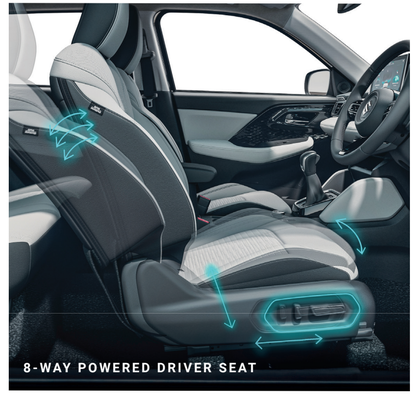
*Advanced Driver Assistance System (ADAS) is not a substitute for human eye and driver vigilance, it is a driver assist system that enhances driving experience and safety. The system performance may vary from road and driving conditions. The driver shall remain responsible for safe, vigilant and attentive driving. Features and specifications may vary model wise.

SMART POWERED TAILGATE WITH GESTURE CONTROL

Hands full? Just a swipe with your foot makes access to the boot effortless.

ALL-IN ON COMFORT

In the Victoris, comfort begins long before the journey does. It starts the moment you settle in. The cool embrace of the ventilated seats. The effortless reach of every control. Every aspect of the Victoris adjusts to your personalised comfort to ensure that the road ahead feels less like travel and more like an experience.



8-WAY POWERED DRIVER SEAT

26.03cm MULTI-INFORMATION DIGITAL CLUSTER

AUTO PURIFY WITH PM2.5 DISPLAY

VENTILATED FRONT SEATS

ALL THE TECH.
ALL FOR YOU.

Experience intuitive tech at its best with the Smartplay Pro X Touchscreen Infotainment System. With all your preferred connectivity features, an extensive app store, and a host of personalization options, the Victoris creates a space that's a reflection of you.

AMBIENT LIGHTING

MOBILE TETHERING FOR INTERNET

AUTO PURIFY WITH PM2.5 DISPLAY

ALEXA AUTO VOICE AI

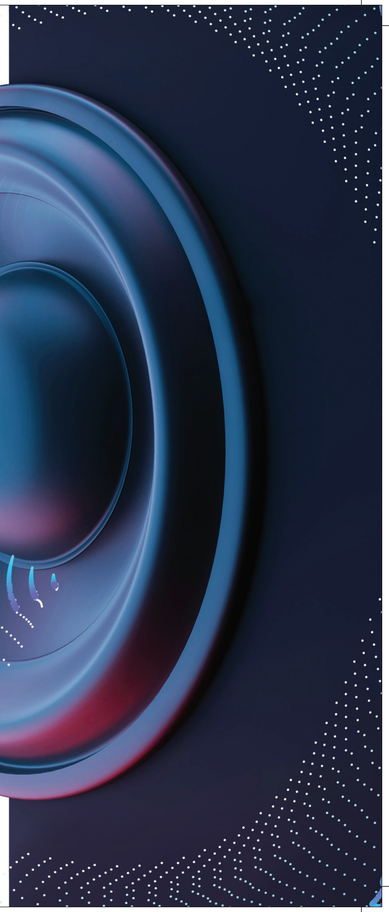
IN-BUILT APP STORE WITH 35+ APPS

ANDROID AUTO™ APPLE CARPLAY™

Infinity by HARMAN | Dolby ATMOS

IMMERSIVE INFINITY SPEAKERS WITH DOLBY ATMOS 5.1 SURROUND SOUND

Every whisper, every raindrop gets its own spotlight with 8 finely-tuned speakers from Infinity by Harman. The immersive 5.1 Channel Dolby Atmos Surround Sound sets the stage for the most vibrant crescendos and dramatic silences. Your drive is where the sound comes alive.



NEXT GEN SUZUKI CONNECT WITH E-CALL

With over 60 smart features, get real-time insights, emergency alerts and vehicle status, on your smartwatch or mobile. The next-gen Suzuki Connect helps you stay connected, informed, and in control, wherever you go.

ALEXA AUTO ON-BOARD

Now take Alexa along for the drive with Alexa Auto Voice AI and over 40 smart features. Stream music, ask questions, and even control smart appliances at home. Experience hands-free convenience, designed for the road.

PERFORMANCE
THAT GOES ALL OUT.

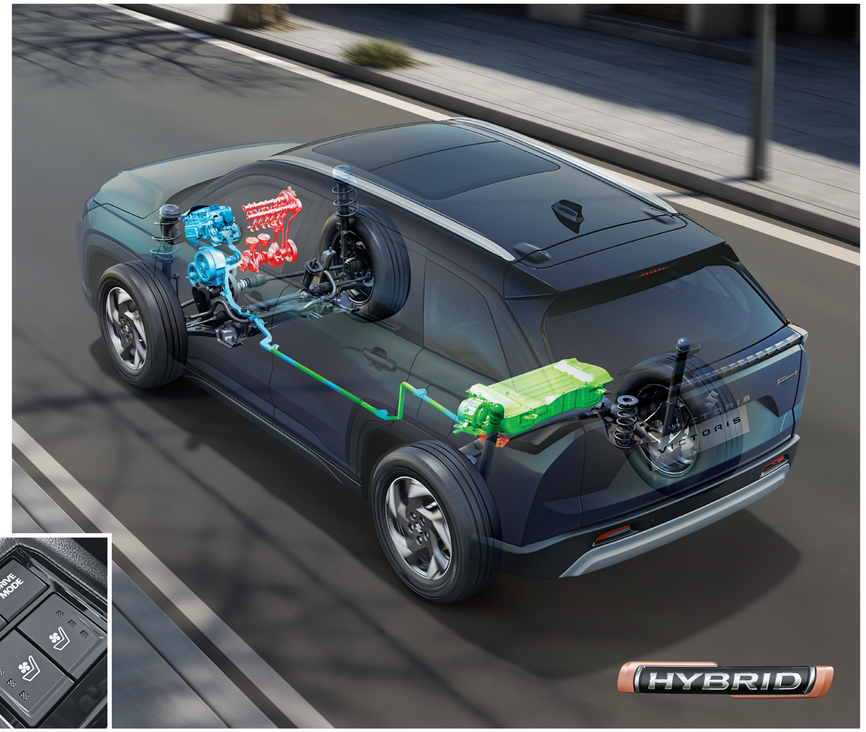
From quick sprints to steady climbs, the Victoris adapts seamlessly to every road and every mood. Agile when the moment calls for it, effortless when the journey demands it. Every drive is tuned to respond to you, delivering exhilaration and calm in equal measure.

AVAILABLE IN:

HYBRID | ALLGRIP | smart HYBRID | S-CNG

FUTURISTIC STRONG HYBRID

For the ones who prefer power without compromise, the Strong Hybrid is the perfect ally. Paired with a battery pack, its refined and silent drive delivers a performance that's smooth, confident, and efficient. With the Victoris, expect nothing less than everything.



ALL-TERRAIN ALLGRIP SELECT

For the ones who own every road they tread, the Victoris with Suzuki ALLGRIP Select is the ultimate companion. From city streets to untamed trails, it adapts seamlessly to challenging terrains, delivering a drive that is as controlled as it is adventurous. Powerful, commanding, and uncompromising, at every turn.

ALL-NEW UNDERBODY S-CNG

A clear boot space with a clean conscience. The Victoris with factory-fitted underbody S-CNG ensures that every journey remains uncompromised. Its clean engine exceeds the fuel efficiency you expect. Its clever design frees-up every inch of your boot space. Its in-built safety features preserve your peace of mind. It's a win-win-win for the Victoris.

THE EXHILARATING SMART HYBRID

The always reliable K15C engine with progressive mild hybrid technology spoils you for choice. Feel every gear shift with a 5-Speed Manual Transmission as it delivers the perfect balance of power and precision for the purist.
6-Speed Automatic Transmission with paddle shifters let you effortlessly switch between control and convenience.
When every shift is smooth, and every acceleration confident, you feel like you own the world - all of it.

ADVANCED PETROL K15C ENGINE
WITH PROGRESSIVE MILD HYBRID

6-SPEED AUTOMATIC TRANSMISSION
WITH PADDLE SHIFTERS

5-SPEED MANUAL TRANSMISSION

BLACK & IVORY INTERIOR WITH SILVER ACCENTS

ALL-BLACK INTERIOR WITH CHAMPAGNE GOLD ACCENTS

GOT IT ALL - IN EVERY COLOUR

The all-new Victoris comes in a variety of stunning hues that are sure to vibe with you.

DUALTONE

ETERNAL BLUE | BLUISH BLACK ROOF

SIZZLING RED | BLUISH BLACK ROOF

SPLENDID SILVER | BLUISH BLACK ROOF

MONOTONE

ETERNAL BLUE

SIZZLING RED

MYSTIC GREEN

BLUISH BLACK

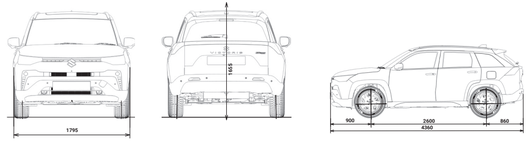
MAGMA GREY

SPLENDID SILVER

ARCTIC WHITE

FEATURE SPECIFICATIONS	PETROL	PETROL/CNG	STRONG HYBRID
Length (mm)	4360		
Width (mm)	1795		
Height (mm)	1655		
Wheelbase (mm)	2600		
Turning Radius (m)	5.4		
Fuel Efficiency	21.18 km/l (MT), 21.06 km/l (AT), 19.07 km/l (ALLGRIP AT)	27.02 km/kg	28.65 km/l (Strong Hybrid Petrol e-CVT)
Fuel Tank Capacity (L)	45	Petrol: 45 CNG: 55 (water eqvt)	45
Seating Capacity (nos)	5		
Kerb Weight (kg)	1145-1195* (MT) 1185-1225* (AT) 1285-1305* (ALLGRIP AT)	1230-1245	1250-1295
Gross Vehicle Weight (kg)	1655(MT), 1685 (AT), 1765 (ALLGRIP AT)	1695	1755
ENGINE			
Engine Type	K15C Smart Hybrid	K15C	M15D
Displacement (cc)	1462		1490
Fuel Type	Petrol	Petrol+CNG	Petrol (Strong Hybrid)
Max Power	75.8 kW (103.06 Ps) @ 6000 rpm	74 kW (100.6 Ps) @ 6000 rpm - Petrol Mode 64.6 kW (87.8 Ps) @ 5500 rpm - CNG Mode	68kW (92.45 Ps) @ 5500 rpm
Max Torque	139 Nm @ 4300 rpm	137.1 Nm @ 4300 rpm - Petrol Mode 121.5 Nm @ 4200 rpm - CNG Mode	122 Nm @ 3800-4800 rpm
HYBRID SYSTEM			
Battery Type	Lithium-ion	-	Lithium-ion
Battery Voltage	-	-	177.6 V
Motor Generator Type	-	-	AC Synchronous Motor
Max Power	-	-	59 kW (80 Ps) @ 3995-5500
Max Torque	-	-	141 Nm @ 0-2695
Driving Mode	-	-	ECO + Normal + Power
ISG	12V - 6Ah (Li-Ion)	-	-
TRANSMISSION			
Type	5MT/6AT	5MT	e-CVT
DRIVE			
Type	2WD/AWD (ALLGRIP SELECT)	2WD	2WD
SUSPENSION			
Front	MacPherson Strut		
Rear	Torsion Beam		
BRAKES			
Front	Disc		
Rear	Disc		
WHEEL TYRE			
Tyre Size	215/60 R17		

POWERTRAIN		Lxi	Vxi	Zxi / Zxi (O)	Zxi+ / Zxi+ (O)
1.5L K15C Petrol ISG	5-speed Manual Transmission (5MT)	✓	✓	✓	✓
	6-speed Automatic Transmission (6AT)	-	✓	✓	✓
	AllGrip Select (6AT)	-	-	-	✓
1.5L K15C Petrol	CNG (5MT)	✓	✓	Zxi only	-
1.5L M15D	Strong Hybrid (eCVT)	-	✓	✓	✓



Drawings shown are general specifications of the vehicle.

FEATURE SPECIFICATIONS					
VARIANTS		**Lxi**	**Vxi**	**Zxi / Zxi (O)**	**Zxi+ / Zxi+ (O)**
EXTERIOR					
Front Headlamps	Projector Headlamps	Halogen	Halogen	LED	LED
	Positioning Lamps	Halogen	Halogen	LED	LED
	LED Daytime Running Lamps - Dual Function	-	-	✓	✓
	Auto Headlamps with Follow-Me-Home Function	-	-	✓	✓
	LED Front Fog Lamps with Extended Lighting	-	-	✓	✓
Rear Tail Lamps	Dynamic Connected LED Tail Lamps	-	✓	✓	✓
	LED Rear Combination Lamp (with LED Indicator)	✓	✓	✓	✓
Wheel Spec	Alloy Wheels	-	-	✓ (Painted - Black)	✓ (Machined)
	Steel Wheels with Full Wheel Covers	✓	✓	-	-
Exterior Specifications	Front Grille Garnish	Chrome	Chrome	Satin Chrome	Satin Chrome
	Skid Plate (Front & Rear)	Silver	Silver	Silver - Petrol, S-CNG Grey - Strong Hybrid	Silver - Petrol, S-CNG Grey - Strong Hybrid
	All Door Glass Black Out Tape	-	-	✓	✓
	Front Door UV Cut Glasses	-	-	✓	✓
	Rear Door Dark Green Glass	-	-	✓	✓
	Body Coloured Outside Door Handles	-	✓	✓	✓
	Body Coloured ORVMs	-	✓	✓	✓
	ORVMs with Turn Indicators	✓	✓	✓	✓
	Roof Rails	-	✓	✓	✓
	Roof & Quarter Spoiler	✓	✓	✓	✓
	Rear Defogger	✓	✓	✓	✓
	Shark Fin Antenna	✓	✓	✓	✓
	Aero Vents & Aero Edges	✓	✓	✓	✓
	Front Variable Intermittent Wipers	-	-	✓	✓
	Rear Window Wiper & Washer	-	-	✓	✓
	LED High Mount Stop Lamp	✓	✓	✓	✓
INTERIOR					
Interior Theme	Dual Tone Interior (Black + Ivory) with Silver Accents	✓	✓	✓(Petrol, S-CNG)	✓(Petrol, S-CNG)
	All Black Interior with Champagne Gold Accents	-	-	Strong Hybrid	Strong Hybrid
	Door Trim & IP Garnish - Piano Black	-	✓	✓	✓
	Chrome Inside Door Handles	-	✓	✓	✓
Upholstery	Soft Touch IP with Premium Stitch (Black & Ivory-Petrol / Black-Strong Hybrid)	-	-	✓	✓
	Plush Leatherette Seats (Black & Ivory-Petrol / Black-Strong Hybrid)	-	-	-	✓
	Door Trim Armrest	Black Fabric	Black Fabric	Ivory PVC + Stitch-Petrol, S-CNG/Black PVC+ Stitch-Strong Hybrid	Ivory PVC+Stitch-Petrol, S-CNG/Black PVC+ Stitch-Strong Hybrid
	Leatherette Steering Wheel	-	-	-	✓
64 Color Custom Ambient Light	Instrument Panel	-	-	-	Backlit pattern
	Door	-	-	-	Spot
In-Cabin Illumination	Glove Box Light	-	✓	✓	✓
	Front Footwell Light (Driver & Co-Driver Side)	-	✓	✓	✓
	Courtesy Lamp	-	-	-	Strong Hybrid
	Trunk/Luggage Room Lamp	-	✓	✓	✓
	Rear Reading Lamps	✓	✓	✓	✓
Multi Information Instrument Cluster	10.66 cm (4.2") TFT	✓	✓	✓	-
	Fully Digital 26.03 cm (10.25") TFT	-	Strong Hybrid	✓	✓
COMFORT & CONVENIENCE					
Smart Powered Tailgate with Gesture Control		-	-	✓	✓
Panoramic Sunroof		-	-	(O)	(O)
Seating	Ventilated Seats (Driver + Co-Driver)	-	-	-	✓
	8- Way Driver Powered Seat	-	-	-	✓
	Seat Height Adjuster	-	✓	✓	✓
	60:40 Folding Rear Seats	✓	✓	✓	✓
	Adjustable Headrests - All Seats	✓	✓	✓	✓
	Front Seat Back Pocket	-	✓	✓	✓
Head Up Display		-	-	-	✓
Wireless Charger with Active Cooling		-	-	✓	✓
Auto Purify with PM 2.5 Display		-	-	-	✓
Charging	USB Charger Rear Type C+C (45 W + 45 W, Max 60 W)	-	✓	✓	✓
	USB Charger Front	✓(Type A)	✓(Type A)	✓(Type A)	✓(Type C-15W + Type C-45W)
	Accessory Socket (Center Console)	✓	✓	✓	✓
	Accessory Socket (Luggage Room)	-	✓	✓	✓
Push Start with Smart Key		-	✓	✓	✓
Auto AC		✓	✓	✓	✓
Cabin Air Filter		Pollen	PM 2.5	PM 2.5	PM 2.5
Rear AC Vents		✓	✓	✓	✓
Auto Folding ORVMs		-	-	-	✓
Electrically Foldable ORVMs		-	✓	✓	✓
Vanity Mirror (Driver + Co-Driver)		✓	✓	✓	✓

FEATURE SPECIFICATIONS					
VARIANTS		**Lxi**	**Vxi**	**Zxi / Zxi (O)**	**Zxi+ / Zxi+ (O)**
COMFORT & CONVENIENCE					
Vanity Mirror Lamp (Driver + Co-Driver)		-	-	-	✓
Front Sliding Armrest (with Storage)		✓	✓	✓	✓
Rear Center Armrest with Cup Holder		✓	✓	✓	✓
Sunglass Holder		-	-	✓	✓
Adjustable Tilt & Telescopic Power Steering		✓	✓	✓	✓
Cruise Control		-	✓	✓	✓(MT & Strong Hybrid)
Keyless Entry		✓	✓	✓	✓
Paddle Shifters		-	6AT	6AT	6AT
Gear Shift Indicator		Petrol MT	Petrol MT	Petrol MT	Petrol MT
Engine Drive Modes - Eco, Normal, Sport		-	Strong Hybrid	Strong Hybrid	Strong Hybrid
Terrain Modes - Snow, Sport, Lock, Auto		-	-	-	ALLGRIP
SAFETY & SECURITY					
Level 2 ADAS	Adaptive Cruise Control with all Speed Following & Stop Keeping & Curve Speed Reduction (ACC)	-	-	-	6AT
	Lane Keep Assist (LKA)	-	-	-	6AT
	Lane Departure Prevention (LDP)	-	-	-	6AT
	Lane Departure Warning (LDW)	-	-	-	6AT
	Vehicle Sway Warning	-	-	-	6AT
	Automatic Emergency Braking (AEB) - Car to Car	-	-	-	6AT
	High Beam Assist (HBA)	-	-	-	6AT
	Rear Cross Traffic Alert (RCTA)	-	-	-	6AT
	Forward Collision Warning (FCW)-Car to Car	-	-	-	6AT
	Blind Spot Monitor (BSM) with Lane Change Alert (LCA)	-	-	-	6AT
Emergency Stop Signal (ESS)		-	-	-	6AT
Electronic Stability Program (ESP®)		✓	✓	✓	✓
Traction Control System (TCS)		✓	✓	✓	✓
Engine Drag Control (EDC)		✓	✓	✓	✓
Hill Hold Assist		✓	✓	✓	✓
ABS with EBD and Brake Assist		✓	✓	✓	✓
Hill Descent Control		-	-	-	ALLGRIP
Electronic Parking Brake with Brake Hold		-	6AT	6AT	6AT
360° HD View Camera with 11 Views		-	-	-	✓
Front Parking Sensors with Auto 360° Camera Activation		-	-	-	✓
Rear Parking Sensors		✓	✓	✓	✓
Reverse Parking Camera		-	✓	✓	✓
Tyre-Pressure Monitoring System (TPMS)		-	✓	✓	✓
6 Airbags (Front, Side and Curtain)		✓	✓	✓	✓
Front Seat Belt Pre-Tensioner with Force Limiters		✓	✓	✓	✓
Front Seat Belt Height Adjuster		✓	✓	✓	✓
All Seats Belts (3-Point ELR) with Seat Belt Reminders (Lamp + Buzzer)		✓	✓	✓	✓
Seat Belt Reminder (All Occupants) - Lamp & Buzzer		✓	✓	✓	✓
ISOFIX Child Seat Anchorages		✓	✓	✓	✓
Engine Immobilizer		✓	✓	✓	✓
Day & Night Adjustable IRVM		Manual	Manual	Auto	Auto
Warning Lamp/ Reminder for (Low Fuel, Door Ajar, Headlamp On)		✓	✓	✓	✓
Speed Warning Buzzer		✓	✓	✓	✓
Tyre Repair Kit		✓	✓	✓	✓
Acoustic Vehicle Alert System (AVAS)		-	Strong Hybrid	Strong Hybrid	Strong Hybrid
INFOTAINMENT SYSTEM					
Infotainment & Sound	SmartPlay Pro 17.78cm (7") Touch Screen	✓	✓	✓	-
	SmartPlay Pro X 25.65cm (10.1") HD Touch Screen	-	-	-	✓
	Infinity Premium Sound 8-Speaker System with Centre Speaker & Subwoofer	-	-	-	✓
	Dolby Atmos Surround Sound Experience	-	-	-	✓
	In built 8-Channel Amplifier with 5.1 Surround Sound	-	-	-	✓
Connectivity	Alexa Auto Voice AI	-	-	-	✓
	Suzuki Connect with OTT Apps	-	-	-	✓
	Suzuki Navigation App with Traffic, Speed and Camera Alerts	-	-	-	✓
	Wireless Android Auto, Apple CarPlay	✓	✓	✓	✓
	OTA Updates	✓	✓	✓	✓
	Internet Connectivity	-	-	✓	✓
	Internet Tethering from Smartphone	-	-	-	✓
	Connected Services Powered by New Gen Suzuki Connect	-	✓	✓	✓
	Onboard Voice Assistant (Hi Suzuki with Barge In Feature)	✓	✓	✓	-
	Remote Control App	✓	✓	✓	✓
Door Speakers		2	2	2	4
Front Tweeters		-	2	2	2
AC Temperature Display & Control on Screen		✓	✓	✓	✓
Steering Mounted Controls		✓	✓	✓	✓
Rotary Volume Control Knob		✓	✓	✓	✓

FEATURE SPECIFICATIONS					
VARIANTS		**Lxi**	**Vxi**	**Zxi / Zxi (O)**	**Zxi+ / Zxi+ (O)**
NEXT GEN SUZUKI CONNECT-ADVANCED TELEMATICS SOLUTION SAFETY & SECURITY					
Emergency Alerts		-	✓	✓	✓
Breakdown Notification		-	✓	✓	✓
Stolen Vehicle Notification and Tracking		-	✓	✓	✓
Tow - Away and Tracking		-	✓	✓	✓
Geo-Fence		-	✓	✓	✓
Time Fence		-	✓	✓	✓
Valet Alert		-	✓	✓	✓
Emergency Calling		-	✓	✓	✓
TRIPS AND DRIVING BEHAVIOR					
Trip Summary		-	✓	✓	✓
Driving Behaviour (Score, Ranking, Comparison, Suggestions)		-	✓	✓	✓
Share Trip History		-	✓	✓	✓
Area Guidance		-	✓	✓	✓
Vehicle Location Sharing		-	✓	✓	✓
Send Coordinates & Trip to Car		-	-	-	✓
ALERTS AND NOTIFICATIONS					
Overspeeding		-	✓	✓	✓
Seatbelt		-	✓	✓	✓
AC Idling		-	✓	✓	✓
Trip (Start &End)		-	✓	✓	✓
Low Fuel		-	✓	✓	✓
Low Range		-	✓	✓	✓
Dashboard View		-	✓	✓	✓
Parking Brake Status		-	6AT	6AT	6AT
PM2.5 Monitoring		-	-	-	✓
Vehicle Servicing-Related		-	✓	✓	✓
REMOTE FUNCTIONS					
Door Lock/Cancel Lock		-	-	✓	✓
Hazard Light On/Off		-	-	✓	✓
Headlight Off		-	-	✓	✓
AC On/Off		-	6AT/eCVT	6AT/eCVT	6AT/eCVT
Schedule AC		-	6AT/eCVT	6AT/eCVT	6AT/eCVT
Alarm		-	✓	✓	✓
Immobilizer Request		-	✓	✓	✓
Vehicle Health		-	✓	✓	✓
Smartwatch Connectivity		-	✓	✓	✓
Alexa Skill Connectivity		-	✓	✓	✓
COLOUR VARIANTS					
Eternal Blue with Bluish Black Roof (Dual Tone)		-	-	✓	✓
Sizzling Red with Bluish Black Roof (Dual Tone)		-	-	✓	✓
Splendid Silver with Bluish Black Roof (Dual Tone)		-	-	✓	✓
Eternal Blue		-	✓	-	-
Sizzling Red		-	✓	-	-
Mystic Green		-	-	✓	✓
Bluish Black		-	-	✓	✓
Splendid Silver		✓	✓	✓	✓
Magma Grey		✓	✓	✓	✓
Pearl Arctic White		✓	✓	✓	✓

Drive with a mileage of:

21.18* km/l Petrol (MT)	21.06* km/l Petrol (AT)	27.02* km/kg CNG	19.07* km/l Petrol (ALLGRIP AT)	28.65* km/l Strong Hybrid Petrol e-CVT

3 Years 100 000 km WARRANTY** EXTENDABLE UPTO 6 YEARS

ELECTRONIC STABILITY PROGRAM

LEGEND: (✓) Available, (-) Not Available. Maruti Suzuki India Limited reserves the right to change without notice, price, colours, equipment specifications, and models and to discontinue models and/or replace any part or accessory with any other product. Accessories and features shown in the pictures may not be a part of standard fitment and may differ according to variant. | Black glass on vehicle is due to lighting effect. | Car colour may vary due to printing on paper. | Any offers made will be subject to availability of the product. For more details, contact your nearest Maruti Suzuki ARENA dealership. | *Fuel efficiency as certified by test agency under Rule 115 of Central Motor Vehicle Rules, 1989. | Remote AC operation is available only in Automatic Transmission variant. Suzuki Connect is subscription-based and a network-dependent telematics solution. For detailed features refer to the Suzuki Connect brochure. | Amazon, Alexa, and all related marks are trademarks of Amazon.com, Inc. or its affiliates. | Android Auto™ extends the Android platform into the car in a way that's purpose-built for driving. It is designed to minimise distraction so you can focus on driving with Google™ services like Google Maps™ and Google Search™. Get the Android Auto app on Google Play™. Android Auto is available in the countries listed at the following link: https://www.android.com/auto/#hl=the-road. Most phones with Android 5.0+ work with Android Auto: https://support.google.com/androidauto/#6140477. Android, Android Auto, Google, Google Play and other marks are trademarks of Google Inc. | Apple and iPhone are trademarks of Apple Inc. registered in the U.S. and other countries | Apple CarPlay is a trademark of Apple Inc., Apple CarPlay is available in the countries listed in the following link: http://www.apple.com/ios/feature-availability/#applecarplay-applecarplay Apple CarPlay is compatible with the iPhone models listed in the following link: https://www.apple.com/ios/carplay | Infinity is the trademark of HARMAN International Industries, Incorporated. | Dolby, Dolby Audio, Dolby Atmos and the double-D symbol are registered trademarks of Dolby Laboratories Licensing Corporation. | Some parts of the leather upholstery may contain man-made material. | For details on functioning of safety features including airbags, kindly refer to the Owner's Manual. | Suzuki Connect (available through phone/smartwatch / Suzuki Connect Skill for Amazon Alexa). | Suzuki Connect activation is necessary for App Store, Alexa Auto, Navigation App & Internet Connectivity to function. After expiration of pre-loaded data pack, customer may tether own mobile hotspot to continue using internet-based services. | If the vehicle's speed exceeds 80 km/h, then the primary level warning will beep twice every minute. And, if the speed goes above 120 km/h, the secondary level warning will keep beeping. If the vehicle's speed slows down to 118 km/h, primary warning will resume. If the vehicle is slowed down to 78 km/h, speed alert warning will stop. This buzzer does not indicate any malfunction in the vehicle. | ESP is the registered trademark of Mercedes-Benz Group AG. | **3 years or 100 000 km warranty – whichever is earlier. 6 years or 160 000 km - whichever is earlier. | Trademarks/logos being displayed belong to the respective owners.

VICTORIS

MaJe

TECHNOLOGIES