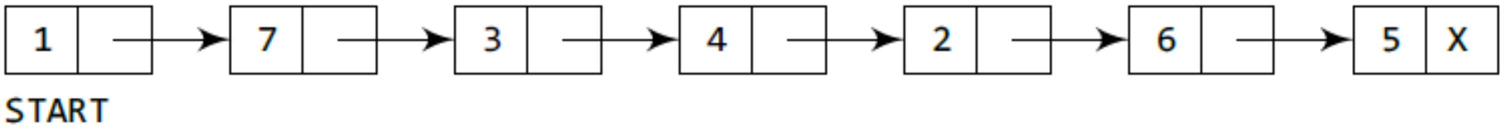
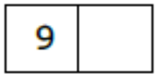
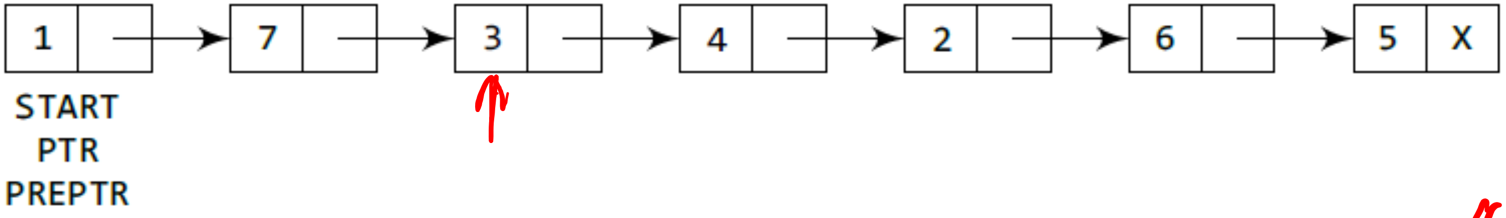


Allocate memory for the new node and initialize its DATA part to 9.



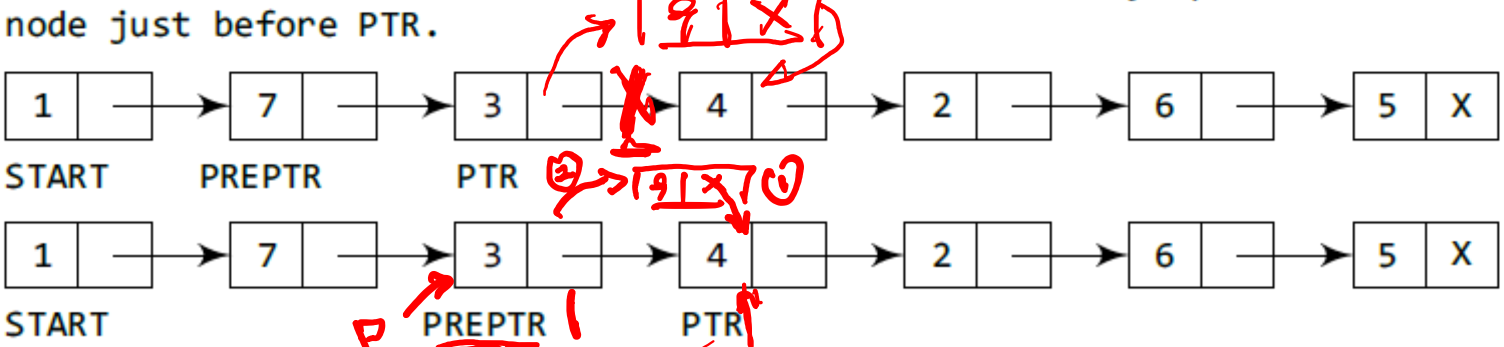
Inserting a Node after Node that has Value NUM (3)

Take two pointer variables PTR and PREPTR and initialize them with START so that START, PTR, and PREPTR point to the first node of the list.

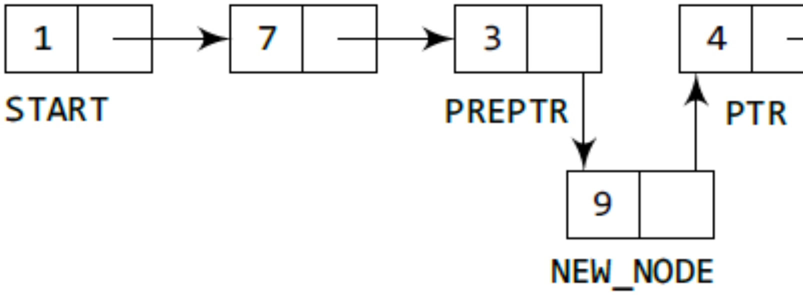


Method 2:
Only one pointer
// p points to given node
newNode->next
= p->next;
p->next = newNode;
find given node

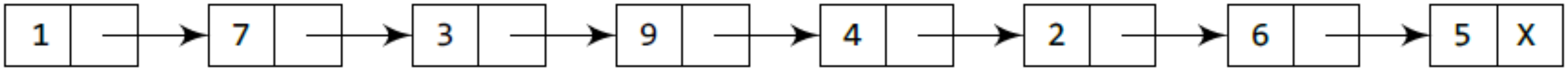
Move PTR and PREPTR until the DATA part of PREPTR = value of the node after which insertion has to be done. PREPTR will always point to the node just before PTR.



Add the new node in between the nodes pointed by PREPTR and PTR.



↘ update/insertion



Insert after a given node (i.e 3).

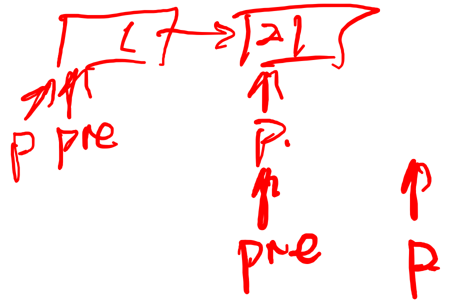
step1: // search the given node

```
Node* p = START;
Node * pre = START;
while ( pre->data != 3 && p != NULL),
{
        pre = p;
        p = p->next;
             NULL.
}.
if (pre->data == 3)  // found
{   // insert
      pre->next = newNode;                      -> O(1).
      newNode ~~=~~> next = p; --> connect the second part.
      return START;
               p = NULL
}
else     // pre->data != 3 or No ~~data~~ give node found,
{
     printf(" Not found given node"); -> O(1)
}    return;
```

if (START == NULL).

pre != NULL && p != NULL).

O(n).



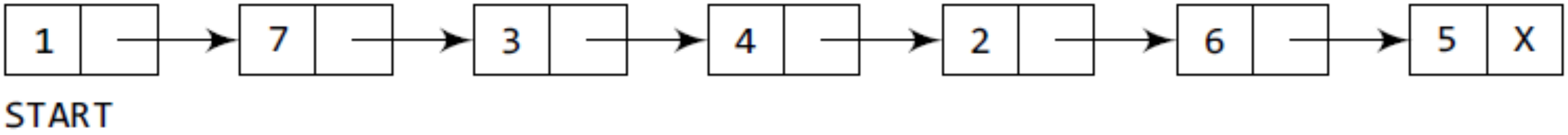
Time Complexity:
O(n).

Inserting a Node after Node that has Value NUM

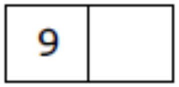
```
ALGORITHM TO INSERT A NEW NODE AFTER A NODE THAT HAS VALUE NUM

Step 1: IF AVAIL = NULL, then
                 Write OVERFLOW
                 Go to Step 12
        [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR
                                                  && PTR != NULL
Step 7: Repeat Steps 8 and 9 while PREPTR->DATA != NUM
Step 8:          SET PREPTR = PTR
Step 9:          SET PTR = PTR->NEXT
         [END OF LOOP]
Step 10: SET PREPTR->NEXT = New_Node
Step 11: SET New_Node->NEXT = PTR
Step 12: EXIT
```

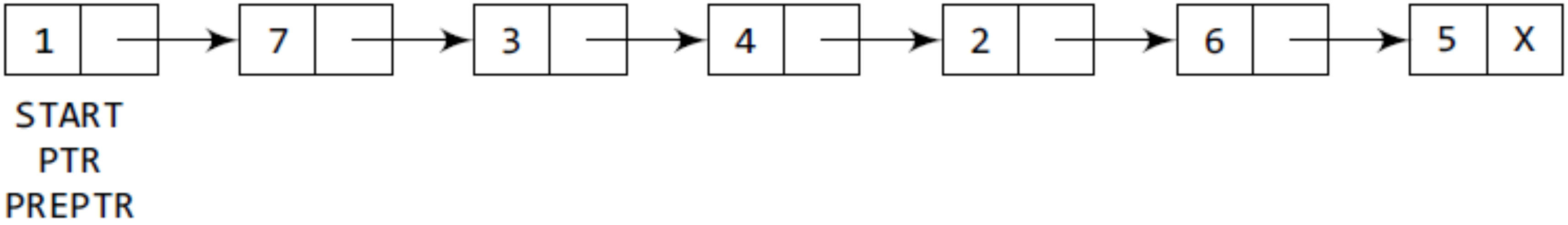
Inserting a Node before Node that has Value NUM (3)



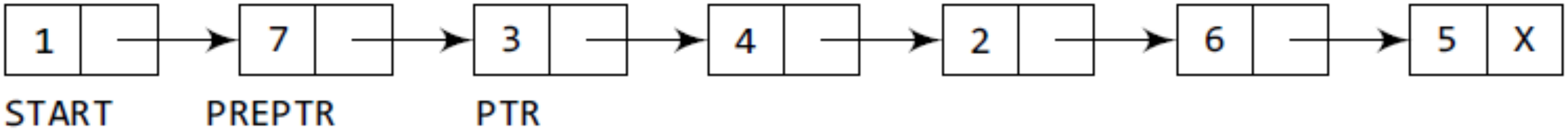
Allocate memory for the new node and initialize its DATA part to 9.



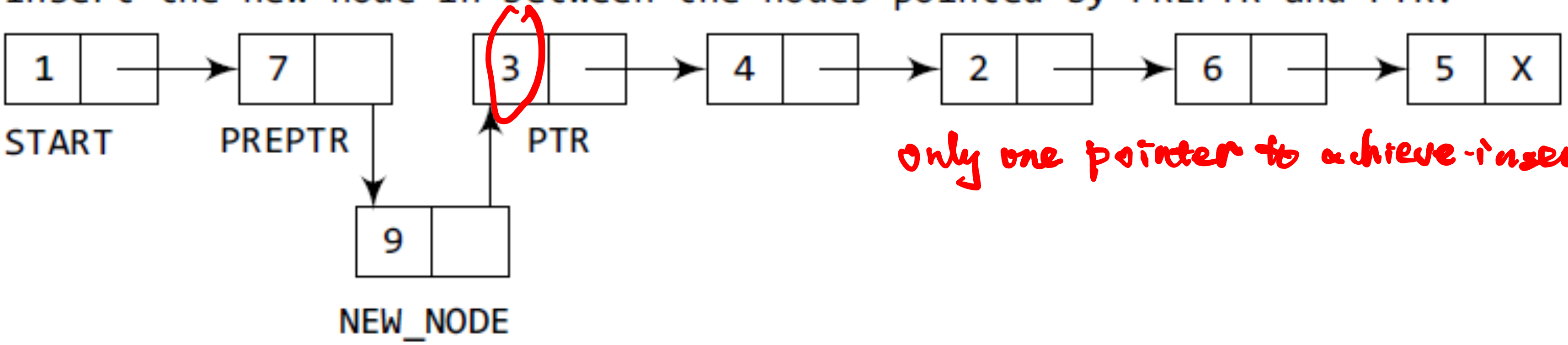
Initialize PREPTR and PTR to the START node.



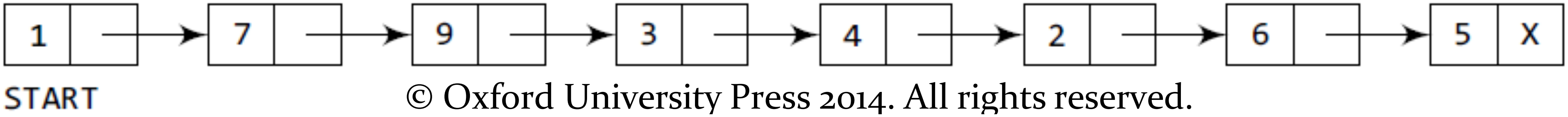
Move PTR and PREPTR until the DATA part of PTR = value of the node before which insertion has to be done. PREPTR will always point to the node just before PTR.



Insert the new node in between the nodes pointed by PREPTR and PTR.



only one pointer to achieve insertion ✓



Inserting a Node Before a Node that has Value NUM

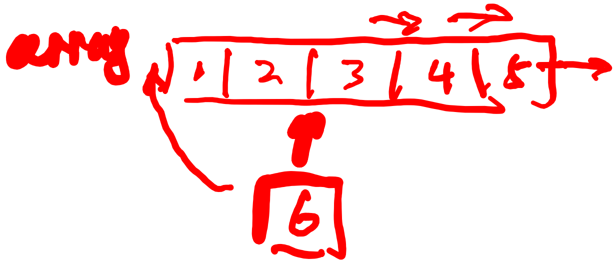
```
ALGORITHM TO INSERT A NEW NODE AFTER A NODE THAT HAS VALUE NUM

Step 1: IF AVAIL = NULL, then
                 Write OVERFLOW
                 Go to Step 12
        [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET PTR = START
Step 6: SET PREPTR = PTR                             && PTR != NULL
Step 7: Repeat Steps 8 and 9 while PTR->DATA != NUM
Step 8:          SET PREPTR = PTR
Step 9:          SET PTR = PTR->NEXT
        [END OF LOOP]
Step 10: SET PREPTR->NEXT = New_Node
Step 11: SET New_Node->NEXT = PTR
Step 12: EXIT
```

Time Complexity – Worst Case

int array[5];
array[6]

	Linked List	Array
Access	O(n)	O(1)
Search	O(n)	O(n)
Insertion	O(1) without traversal	O(n)



worst case: O(n)
best case: O(1).

Deleting a Node from a Linked List

- Three cases
  - Case 1: The first node is deleted.
  - Case 2: The last node is deleted.
  - Case 3: The node after a given node is deleted.

free (ptr) //return to free Memory

ptr = NULL,

1st step

- Underflow is a condition that occurs when we try to delete a node from a linked list that is empty. This happens when START = NULL or when there are no more nodes to delete.

After deletion.

- Note that when we delete a node from a linked list, we actually have to free the memory occupied by that node.
  - The memory is returned to the free pool so that it can be used to store other programs and data.
- Whatever be the case of deletion, we always change the AVAIL pointer so that it points to the address that has been recently vacated.

Deleting the First Node

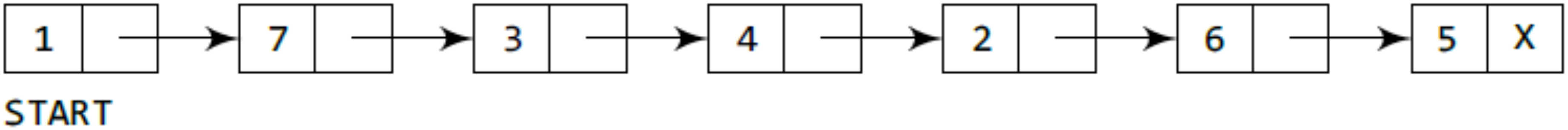
```
Algorithm to delete the first node from the linked list

Step 1: IF START = NULL, then
                 Write UNDERFLOW
                 Go to Step 5
          [END OF IF]
Step 2: SET PTR = START
Step 3: SET START = START->NEXT
Step 4: FREE PTR
Step 5: EXIT
```

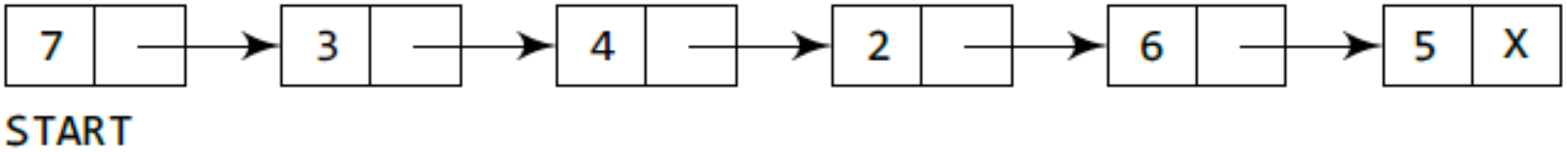
Time complexity.
O(1).

delete 1st node.

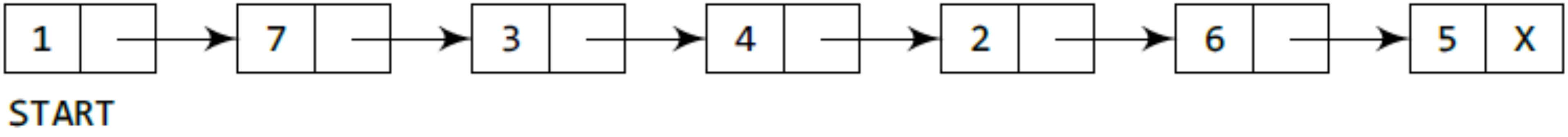
free(ptr);
ptr = NULL;



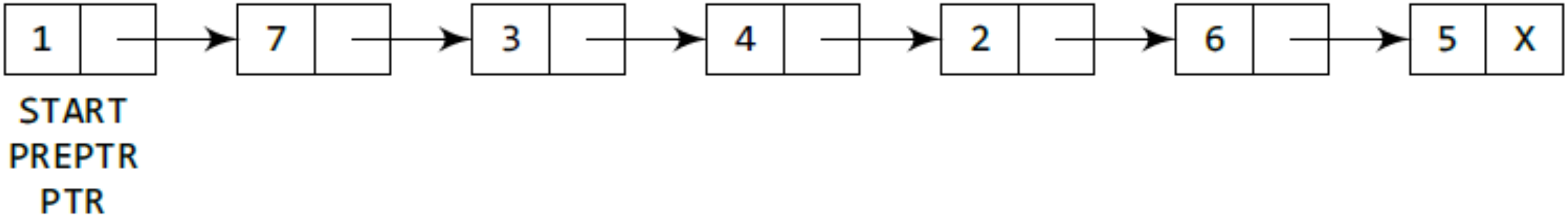
Make START to point to the next node in sequence.



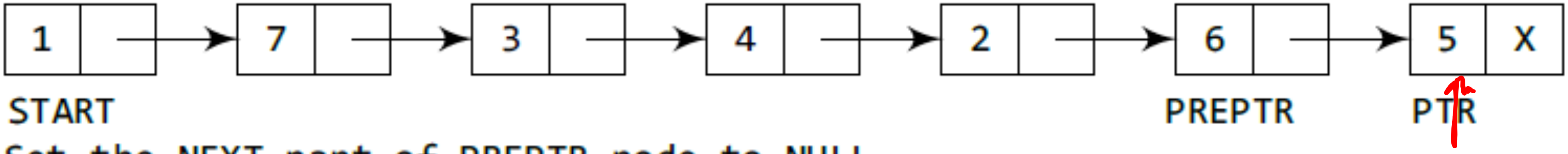
Deleting the Last Node



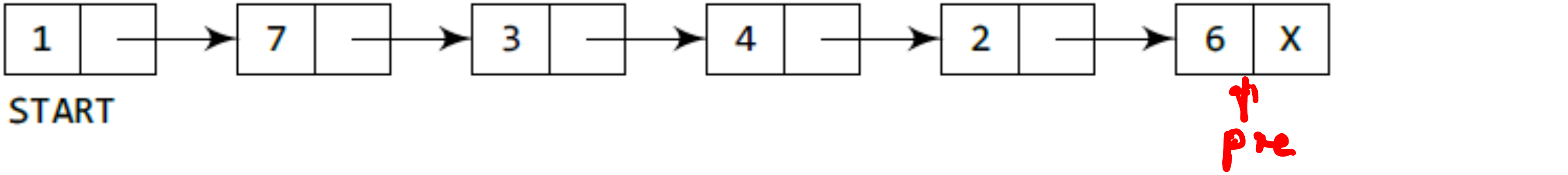
Take pointer variables PTR and PREPTR which initially point to START.



Move PTR and PREPTR such that NEXT part of PTR = NULL. PREPTR always points to the node just before the node pointed by PTR.



Set the NEXT part of PREPTR node to NULL.



Deleting the Last Node

```
ALGORITHM TO DELETE THE LAST NODE OF THE LINKED LIST

Step 1: IF START = NULL, then
                Write UNDERFLOW
                Go to Step 8
        [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR->NEXT != NULL
Step 4:         SET PREPTR = PTR
Step 5:         SET PTR = PTR->NEXT
        [END OF LOOP]
Step 6: SET PREPTR->NEXT = NULL
Step 7: FREE PTR
Step 8: EXIT
```

underflow,

pre→

PTR

fre (prePTR)

prePTR=NULL

move pre, p → node by node.

// deletion

→ free memory free( )

PTR = NULL

C Programming Tools & Resources

1. Replit – Online C Compiler

🔷 **Run & test C programs directly in your browser**

🔷 No installation required, great for quick prototyping

🔷 Try it here: **Replit C Compiler**

2. Visual Studio Code (VS Code)

🔷 **Lightweight, versatile code editor with extensions**

🔷 Supports debugging, syntax highlighting, and Git integration

🔷 Download VS Code

🔷 Getting Started Guide

3. AI Assistance & GitHub Copilot

🔷 **Best Practice:** Write your code first, then use AI for improvements

🔷 Helps with syntax, debugging, and code optimization

📌 *Tip:* AI tools are helpful but should **not replace** learning and problem-solving skills!