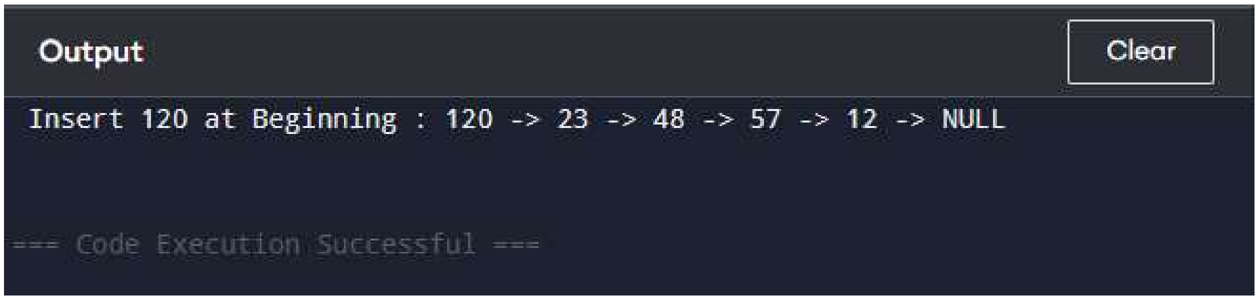
1.SLL Inserting 120 at Beginning:

```
#include <stdio.h>
#include <stdlib.h>


struct Node {
    int data;
    struct Node* next;
};


struct Node* insertAtBeginning(struct Node* head, int newData) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        return head;
    }

    newNode->data = newData;
    newNode->next = head;


    head = newNode;

    return head;
}


void printList(struct Node* head) {
    struct Node* temp = head;
     printf(" Insert 120 at Beginning : ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```

```
int main() {

    struct Node* head = NULL;


    head = insertAtBeginning(head, 12);
    head = insertAtBeginning(head, 57);
    head = insertAtBeginning(head, 48);
    head = insertAtBeginning(head, 23);


    head = insertAtBeginning(head, 120);


    printList(head);

    return 0;
}
```

Output:



```
Insert 120 at Beginning : 120 -> 23 -> 48 -> 57 -> 12 -> NULL


=== Code Execution Successful ===
```

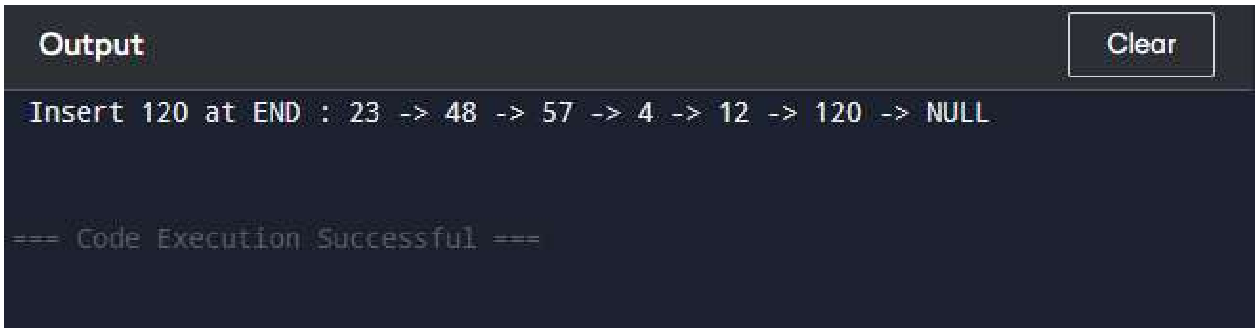
2. SLL Insert 120 at END :

```
#include <stdio.h>
#include <stdlib.h>


struct Node {
    int data;
    struct Node* next;
};
```

```
struct Node* insertAtEnd(struct Node* head, int newData) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        return head;
    }


    newNode->data = newData;
    newNode->next = NULL;


    if (head == NULL) {
        return newNode;
    }


    struct Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }


    temp->next = newNode;

    return head;
}


void printList(struct Node* head) {
    struct Node* temp = head;
     printf(" Insert 120 at END : ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```

```
int main() {

    struct Node* head = NULL;


    head = insertAtEnd(head, 23);
    head = insertAtEnd(head, 48);
    head = insertAtEnd(head, 57);
    head = insertAtEnd(head, 4);
    head = insertAtEnd(head, 12);


    head = insertAtEnd(head, 120);


    printList(head);

    return 0;
}
```

Output:



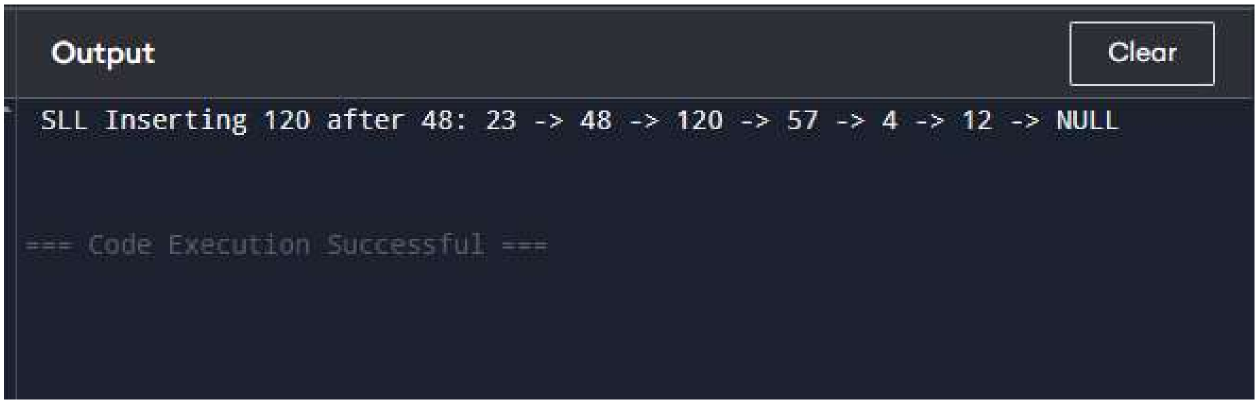
```
Insert 120 at END : 23 -> 48 -> 57 -> 4 -> 12 -> 120 -> NULL


=== Code Execution Successful ===
```

3. SLL Inserting 120 after 48:

```
#include <stdio.h>
#include <stdlib.h>


struct Node {
    int data;
    struct Node* next;
};
```

```
struct Node* insertAfter(struct Node* head, int key, int newData) {
    struct Node* temp = head;


    while (temp != NULL && temp->data != key) {
        temp = temp->next;
    }


    if (temp == NULL) {
        printf("Node with value %d not found.\n", key);
        return head;
    }


    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        return head;
    }


    newNode->data = newData;
    newNode->next = temp->next;
    temp->next = newNode;

    return head;
}


struct Node* insertAtEnd(struct Node* head, int newData) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        return head;
    }
    newNode->data = newData;
    newNode->next = NULL;
    if (head == NULL) {
        return newNode;
    }

    struct Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;

    return head;
}


void printList(struct Node* head) {
    struct Node* temp = head;
     printf(" SLL Inserting 120 after 48: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {

    struct Node* head = NULL;


    head = insertAtEnd(head, 23);
    head = insertAtEnd(head, 48);
    head = insertAtEnd(head, 57);
    head = insertAtEnd(head, 4);
    head = insertAtEnd(head, 12);


    head = insertAfter(head, 48, 120);

    printList(head);
    return 0;
}
```

Output:

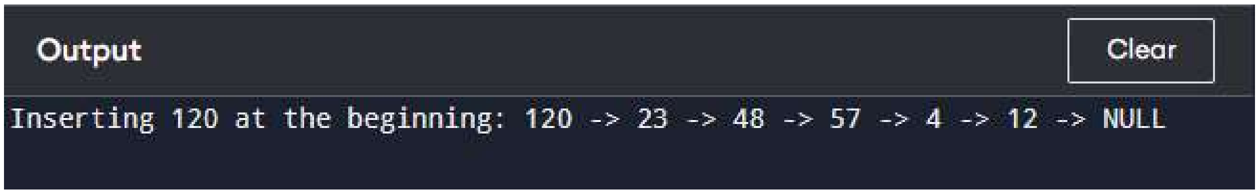


```
SLL Inserting 120 after 48: 23 -> 48 -> 120 -> 57 -> 4 -> 12 -> NULL


=== Code Execution Successful ===
```

4. DLL Inserting 120 at the beginning:

```
#include <stdio.h>
#include <stdlib.h>


struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};


void insertAtBeginning(struct Node** head, int value) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->prev = NULL;


    if (*head == NULL) {
        newNode->next = NULL;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
    }


    *head = newNode;
}


void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;


    insertAtBeginning(&head, 12);
    insertAtBeginning(&head, 4);
    insertAtBeginning(&head, 57);
    insertAtBeginning(&head, 48);
    insertAtBeginning(&head, 23);


    insertAtBeginning(&head, 120);


    printf("Inserting 120 at the beginning: ");
    printList(head);

    return 0;
}
```

Output

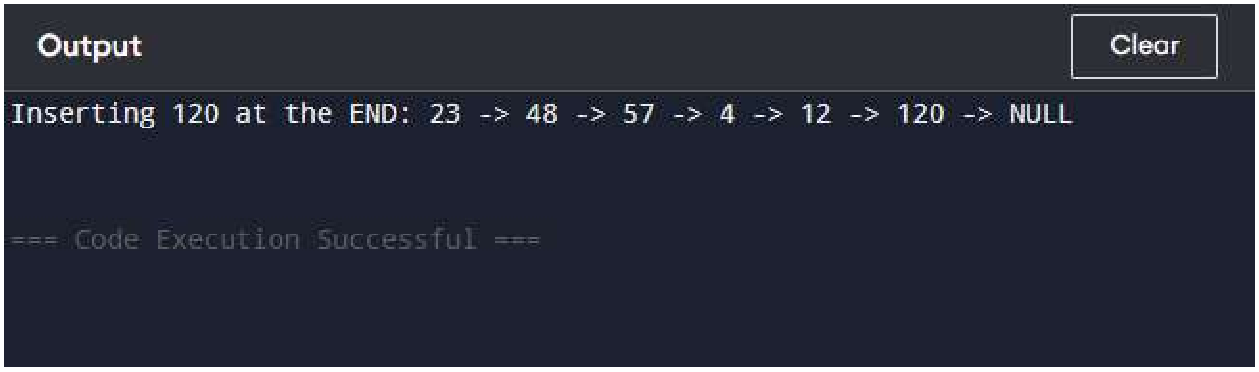


```
Inserting 120 at the beginning: 120 -> 23 -> 48 -> 57 -> 4 -> 12 -> NULL
```

5. DLL Inserting 120 at the END:

```
#include <stdio.h>
#include <stdlib.h>


struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};


void insertAtEnd(struct Node** head, int value) {

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;


    if (*head == NULL) {
        newNode->prev = NULL;
        *head = newNode;
        return;
    }


    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }


    temp->next = newNode;
    newNode->prev = temp;
}


void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;


    insertAtEnd(&head, 23);
    insertAtEnd(&head, 48);
    insertAtEnd(&head, 57);
    insertAtEnd(&head, 4);
    insertAtEnd(&head, 12);


    insertAtEnd(&head, 120);


    printf("Inserting 120 at the END: ");
    printList(head);

    return 0;
}
```

Output



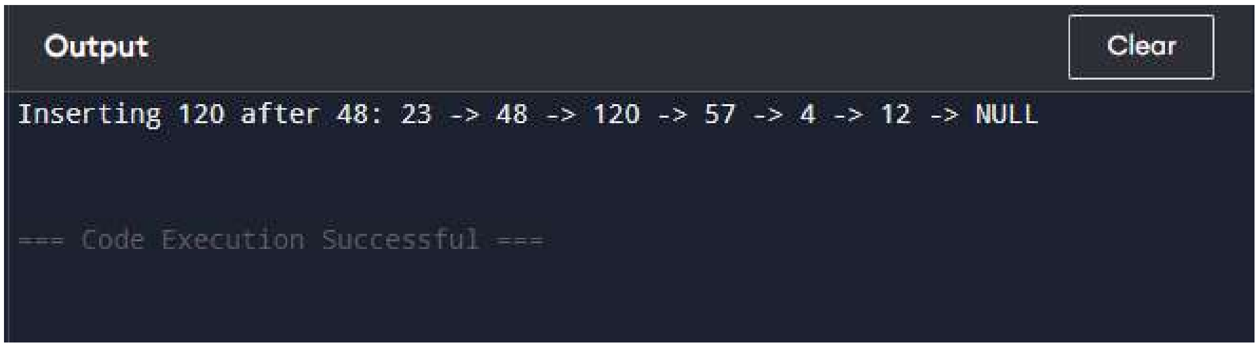
```
Inserting 120 at the END: 23 -> 48 -> 57 -> 4 -> 12 -> 120 -> NULL


=== Code Execution Successful ===
```

6. DLL Inserting 120 after 48:

```
#include <stdio.h>
#include <stdlib.h>


struct Node {
   int data;
   struct Node* next;
   struct Node* prev;
};


void insertAtEnd(struct Node** head, int value) {

   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = value;
   newNode->next = NULL;


   if (*head == NULL) {
      newNode->prev = NULL;
      *head = newNode;
```

```
        return;
    }


    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }


    temp->next = newNode;
    newNode->prev = temp;
}


void insertAfterValue(struct Node** head, int afterValue, int newValue) {
    struct Node* temp = *head;


    while (temp != NULL && temp->data != afterValue) {
        temp = temp->next;
    }


    if (temp != NULL) {
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = newValue;


        newNode->next = temp->next;
        newNode->prev = temp;
        if (temp->next != NULL) {
            temp->next->prev = newNode;
        }
        temp->next = newNode;
    }
}


void printList(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main() {
    struct Node* head = NULL;


    insertAtEnd(&head, 23);
    insertAtEnd(&head, 48);
    insertAtEnd(&head, 57);
    insertAtEnd(&head, 4);
    insertAtEnd(&head, 12);


    insertAfterValue(&head, 48, 120);


    printf("Doubly Linked List after inserting 120 after 48: ");
    printList(head);

    return 0;
}
```

Output



```
Inserting 120 after 48: 23 -> 48 -> 120 -> 57 -> 4 -> 12 -> NULL


=== Code Execution Successful ===
```