main.c

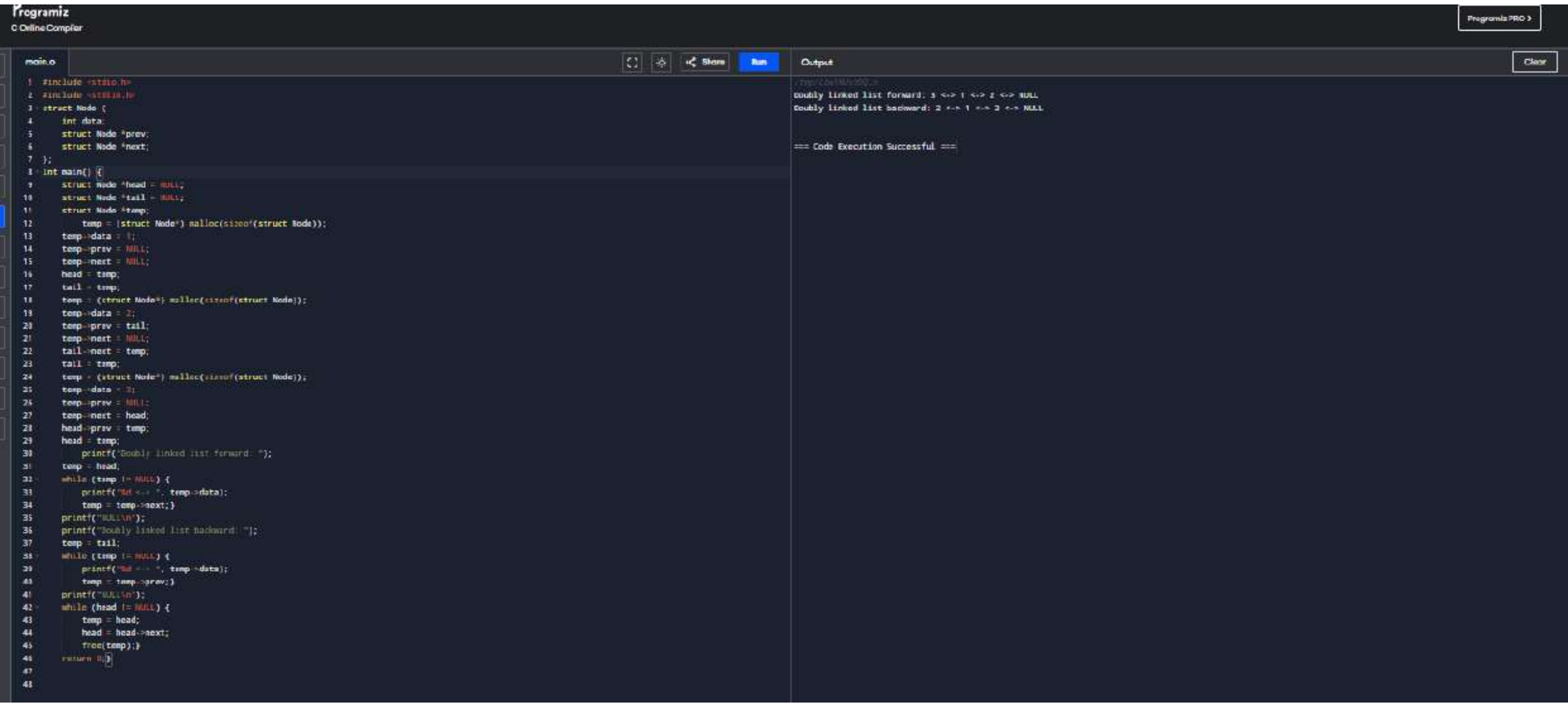
Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next; };
int main() {
    struct Node *head = NULL;
    struct Node *temp;
    head = (struct Node*) malloc(sizeof(struct Node));
    head->data = 1;
    head->next = NULL;
    temp = (struct Node*) malloc(sizeof(struct Node));
    temp->data = 2;
    temp->next = head;
    head = temp;
    temp = (struct Node*) malloc(sizeof(struct Node));
    temp->data = 3;
    temp->next = head;
    head = temp;
    printf("Linked list: ");
    temp = head;
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;}
    printf("NULL\n");
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);}
    return 0;}
```

Output | Clear

```
/tmp/AYyMqtyEbG.o
Linked list: 3 -> 2 -> 1 -> NULL


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



Programiz
C Online Compiler
Programiz PRO >
main.c
Share
Run
Output
Clear
#include <stdio.h>
#include <stdlib.h>
struct Node {
int data;
struct Node *prev;
struct Node *next;
};
int main() {
struct Node *head = NULL;
struct Node *tail = NULL;
struct Node *temp;
temp = (struct Node*) malloc(sizeof(struct Node));
temp->data = 1;
temp->prev = NULL;
temp->next = NULL;
head = temp;
tail = temp;
temp = (struct Node*) malloc(sizeof(struct Node));
temp->data = 2;
temp->prev = tail;
temp->next = NULL;
tail->next = temp;
tail = temp;
temp = (struct Node*) malloc(sizeof(struct Node));
temp->data = 3;
temp->prev = NULL;
temp->next = head;
head->prev = temp;
head = temp;
printf("Doubly linked list forward: ");
temp = head;
while (temp != NULL) {
printf("%d <-> ", temp->data);
temp = temp->next;}
printf("NULL\n");
printf("Doubly linked list backward: ");
temp = tail;
while (temp != NULL) {
printf("%d <-> ", temp->data);
temp = temp->prev;}
printf("NULL\n");
while (head != NULL) {
temp = head;
head = head->next;
free(temp);}
return 0;}
Doubly linked list forward: 3 <-> 1 <-> 2 <-> NULL
Doubly linked list backward: 2 <-> 1 <-> 3 <-> NULL
=== Code Execution Successful ===

Programiz

C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
int main() {
    struct Node *head = NULL;
    struct Node *temp;
        head = (struct Node*) malloc(sizeof(struct Node));
    head->data = 1;
    head->next = head;
        temp = (struct Node*) malloc(sizeof(struct Node));
    temp->data = 2;
    temp->next = head->next;
    head->next = temp;
    temp = (struct Node*) malloc(sizeof(struct Node));
    temp->data = 3;
    temp->next = head->next->next;
    head->next->next = temp;
    printf("Circular linked list: ");
    temp = head;
    do {
        printf("%d -> ", temp->data);
        temp = temp->next;
    } while (temp != head);
    printf("%d\n", temp->data);
    struct Node *current = head;
    struct Node *next;
    do {
        next = current->next;
        free(current);
        current = next;
    } while (current != head);
    return 0;
}
```

Output

Clear

```
/tmp/AQ6uOY2k6c.o
Circular linked list: 1 -> 2 -> 3 -> 1


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

Programiz

C Online Compiler

Programiz PRO

main.c

Share Run

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX_SIZE 100
int stack[MAX_SIZE];
int top = -1;
int isFull() {
    return top == MAX_SIZE - 1;
}
int isEmpty() {
    return top == -1;
}
void push(int data) {
    if (isFull()) {
        printf("Stack overflow!\n");
        return;
    }
    stack[++top] = data;
    printf("Pushed %d to stack\n", data);
}
void pop() {
    if (isEmpty()) {
        printf("Stack underflow!\n");
        return;
    }
    printf("Popped %d from stack\n", stack[top--]);
}
int peek() {
    if (isEmpty()) {
        printf("Stack is empty!\n");
        return -1;
    }
    return stack[top];
}
int main() {
    push(1);
    push(2);
    push(3);
        printf("Top element of stack: %d\n", peek());
        pop();
    pop();
    pop();
    pop();
    return 0;
}
```

Output

Clear

```
Pushed 1 to stack
Pushed 2 to stack
Pushed 3 to stack
Top element of stack: 3
Popped 3 from stack
Popped 2 from stack
Popped 1 from stack
Stack underflow!


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

Programiz
C Online Compiler

Programiz PRO >

main.c

Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
struct Node *top = NULL;
int isEmpty() {
    return top == NULL;
}
void push(int data) {
    struct Node *newNode = (struct Node*) malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        return;
    }
    newNode->data = data;
    newNode->next = top;
    top = newNode;
    printf("Pushed %d to stack\n", data);
}
void pop() {
    if (isEmpty()) {
        printf("Stack underflow!\n");
        return;
    }
    struct Node *temp = top;
    printf("Popped %d from stack\n", temp->data);
    top = top->next;
    free(temp);
}
int peek() {
    if (isEmpty()) {
        printf("Stack is empty!\n");
        return -1; // or any sentinel value
    }
    return top->data;
}
int main() {
    push(1);
    push(2);
    push(3);
    printf("Top element of stack: %d\n", peek());
    pop();
    pop();
    pop();
    pop();
    return 0;
}

```

Output

Clear

```
Pushed 1 to stack
Pushed 2 to stack
Pushed 3 to stack
Top element of stack: 3
Popped 3 from stack
Popped 2 from stack
Popped 1 from stack
Stack underflow!


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

programiz.com/c-programming/online-compiler/

Programiz
C Online Compiler

main.c

Share | Run

```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node *next;
};
void insertAtBeginning(struct Node **head_ref, int new_data) {
    struct Node *new_node = (struct Node *)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = *head_ref;
    *head_ref = new_node;}
void printList(struct Node *node) {
    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;}
    printf("\n");}
int main() {
    struct Node *head = NULL;
        insertAtBeginning(&head, 1);
    insertAtBeginning(&head, 2);
    insertAtBeginning(&head, 3);
    printf("Linked list: ");
    printList(head);
    return 0;}
```

Output | Clear

```
/tmp/YxDedQDxWN.o
Linked list: 3 2 1


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