NAME: JERRY DAVID R (192424401)

COURSE NAME: DATA STRUCTURES FOR MODERN COMPUTING SYSTEMS

COURSE CODE: CSA0302

Experiment 14: Double Linked List

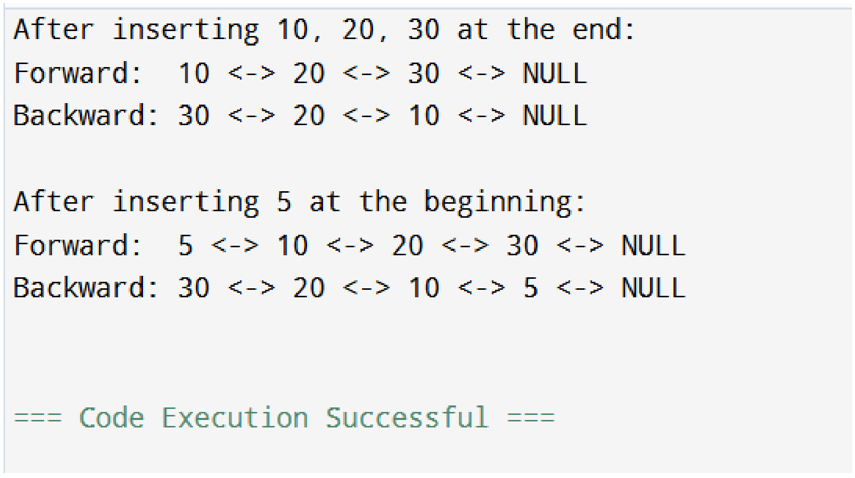
Code:

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* next;
   struct Node* prev;
};
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if (newNode == NULL) {
      printf("Memory allocation failed!\n");
      exit(1);
   }
   newNode->data = data;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}
void insertAtBeginning(struct Node** headRef, int data) {
   struct Node* newNode = createNode(data);
   newNode->next = *headRef;
   if (*headRef != NULL) {
      (*headRef)->prev = newNode;
   }
   *headRef = newNode;
```

```
}
void insertAtEnd(struct Node** headRef, int data) {
    struct Node* newNode = createNode(data);
    if (*headRef == NULL) {
        *headRef = newNode;
        return;
    }
    struct Node* last = *headRef;
    while (last->next != NULL) {
        last = last->next;
    }
    last->next = newNode;
    newNode->prev = last;
}
void printListForward(struct Node* node) {
    printf("Forward:  ");
    while (node != NULL) {
        printf("%d <-> ", node->data);
        node = node->next;
    }
    printf("NULL\n");
}
void printListBackward(struct Node* head) {
    if (head == NULL) return;
    struct Node* last = head;
    while (last->next != NULL) {
        last = last->next;
    }
    printf("Backward: ");
    while (last != NULL) {
        printf("%d <-> ", last->data);
        last = last->prev;
    }
    printf("NULL\n");
}
void freeList(struct Node* head) {
   struct Node* tmp;
   while (head != NULL) {
       tmp = head;
       head = head->next;
       free(tmp);
    }
}
int main() {
    struct Node* head = NULL;
    insertAtEnd(&head, 10);
    insertAtEnd(&head, 20);
    insertAtEnd(&head, 30);
    printf("After inserting 10, 20, 30 at the end:\n");
    printListForward(head);
    printListBackward(head);
    printf("\n");
    insertAtBeginning(&head, 5);
    printf("After inserting 5 at the beginning:\n");
    printListForward(head);
    printListBackward(head);

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

Output:


```
After inserting 10, 20, 30 at the end:
Forward:  10 <-> 20 <-> 30 <-> NULL
Backward: 30 <-> 20 <-> 10 <-> NULL

After inserting 5 at the beginning:
Forward:  5 <-> 10 <-> 20 <-> 30 <-> NULL
Backward: 30 <-> 20 <-> 10 <-> 5 <-> NULL



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