Linear Queue operations using an array

Code:

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

void push(struct Node** top, int data) {
   struct Node* new_node = newNode(data);
   new_node->next = *top;
   *top = new_node;
   printf("%d pushed to stack\n", data);
}

int pop(struct Node** top) {
   if (*top == NULL) {
      printf("Stack is empty\n");
      return -1; // Error condition
   }
   struct Node* temp = *top;
   int popped = temp->data;
   *top = temp->next;
   free(temp);
   return popped;
}

void display(struct Node* top) {
   if (top == NULL) {
      printf("Stack is empty\n");
      return;
   }
   printf("Stack: ");
   while (top != NULL) {
      printf("%d ", top->data);
      top = top->next;
   }
   printf("\n");
}

int main() {
   struct Node* top = NULL;
   int choice, data;

   while (1) {
      printf("\n1. Push\n2. Pop\n3. Display\n4. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
            printf("Enter data to push: ");
            scanf("%d", &data);
            push(&top, data);
            break;
         case 2:
            data = pop(&top);
            if (data != -1) {
```

```
            printf("%d popped from stack\n", data);
        }
        break;
      case 3:
        display(top);
        break;
      case 4:
        exit(0);
      default:
        printf("Invalid choice\n");
    }
  }

  return 0;
}
```

OUTPUT:

```
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter data to push: 1
1 pushed to stack

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter data to push: 2
2 pushed to stack

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 1
Enter data to push: 3
3 pushed to stack

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack: 3 2 1

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
3 popped from stack

1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Stack: 2 1
```