createList(int n) : Node

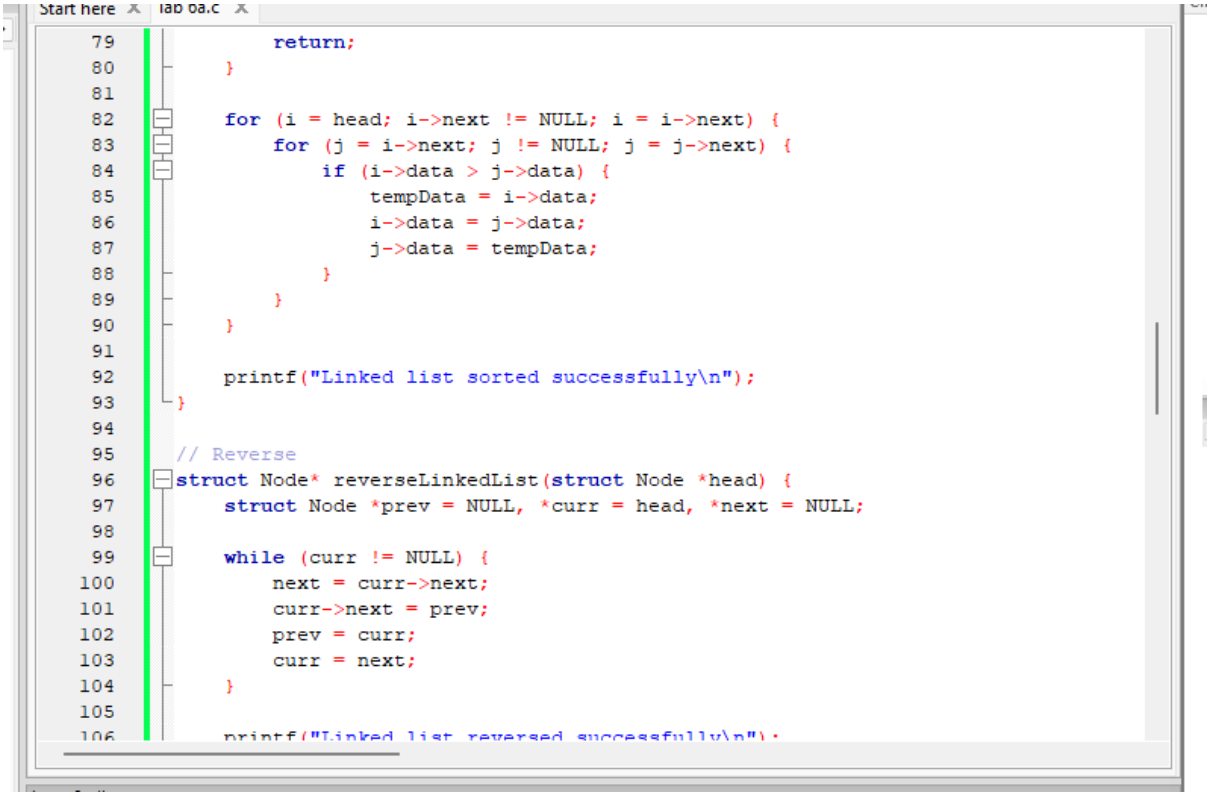
Start here | lab 6a.c

```c
    }

    for (int i = 1; i <= n; i++) {
        newNode = (struct Node*)malloc(sizeof(struct Node));
        if (newNode == NULL) {
            printf("Memory allocation failed\n");
            return head;
        }

        printf("Enter data for node %d: ", i);
        scanf("%d", &data);

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

        if (head == NULL)
            head = newNode;
        else
            temp->next = newNode;

        temp = newNode;
    }

    printf("Linked list created successfully\n");
    return head;
}

// Display
```

Start here | lab 6a.c

```c
    return head;
}

// Display
void displayList(struct Node *head) {
    struct Node *temp = head;

    if (head == NULL) {
        printf("List is empty\n");
        return;
    }

    printf("Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

// Sort
void sortLinkedList(struct Node *head) {
    struct Node *i, *j;
    int tempData;

    if (head == NULL) {
        printf("List is empty, cannot sort.\n");
        return;
```

Logs & others


Start here
lab 8a.c
79 return;
80 }
81
82 for (i = head; i->next != NULL; i = i->next) {
83 for (j = i->next; j != NULL; j = j->next) {
84 if (i->data > j->data) {
85 tempData = i->data;
86 i->data = j->data;
87 j->data = tempData;
88 }
89 }
90 }
91
92 printf("Linked list sorted successfully\n");
93 }
94
95 // Reverse
96 struct Node* reverseLinkedList(struct Node *head) {
97 struct Node *prev = NULL, *curr = head, *next = NULL;
98
99 while (curr != NULL) {
100 next = curr->next;
101 curr->next = prev;
102 prev = curr;
103 curr = next;
104 }
105
106 printf("Linked list reversed successfully\n");

```
106     printf("Linked list reversed successfully\n");
107     return prev;
108 }
109
110 // Concatenate
111 struct Node* concatenateLinkedList(struct Node *head1, struct Node *head2) {
112     struct Node *temp;
113
114     if (head1 == NULL)
115         return head2;
116
117     temp = head1;
118     while (temp->next != NULL)
119         temp = temp->next;
120
121     temp->next = head2;
122
123     printf("Linked lists concatenated successfully\n");
124     return head1;
125 }
126
127 // Menu
128 int main() {
129     int choice, n;
130
131     while (1) {
132         printf("\n================================\n");
133         printf("          LINKED LIST MENU      \n");
```

createList(int n) : Node

Start here | lab 6a.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node {
5      int data;
6      struct Node *next;
7  };
8
9  // Global heads
10 struct Node *head1 = NULL;
11 struct Node *head2 = NULL;
12
13 // Function prototypes
14 struct Node* createList(int n);
15 void displayList(struct Node *head);
16 void sortLinkedList(struct Node *head);
17 struct Node* reverseLinkedList(struct Node *head);
18 struct Node* concatenateLinkedList(struct Node *head1, struct Node *head2);
19
20 // Create list
21 struct Node* createList(int n) {
22     struct Node *head = NULL, *newNode, *temp;
23     int data;
24
25     if (n <= 0) {
26         printf("Number of nodes should be greater than 0\n");
27         return NULL;
28     }
```

```
133         printf("        LINKED LIST MENU       \n");
134         printf("==============================\n");
135         printf("1. Create List 1\n");
136         printf("2. Create List 2\n");
137         printf("3. Display List 1\n");
138         printf("4. Display List 2\n");
139         printf("5. Sort List 1\n");
140         printf("6. Reverse List 1\n");
141         printf("7. Concatenate List 1 + List 2\n");
142         printf("8. Exit\n");
143         printf("Enter choice: ");
144         scanf("%d", &choice);
145
146         switch (choice) {
147         case 1:
148             printf("Enter number of nodes for List 1: ");
149             scanf("%d", &n);
150             head1 = createList(n);
151             break;
152
153         case 2:
154             printf("Enter number of nodes for List 2: ");
155             scanf("%d", &n);
156             head2 = createList(n);
157             break;
158
159         case 3:
160             displayList(head1);
```

Start here | lab 6a.c

```
160             displayList(head1);
161             break;
162
163         case 4:
164             displayList(head2);
165             break;
166
167         case 5:
168             sortLinkedList(head1);
169             break;
170
171         case 6:
172             head1 = reverseLinkedList(head1);
173             break;
174
175         case 7:
176             head1 = concatenateLinkedList(head1, head2);
177             printf("After concatenation:\n");
178             displayList(head1);
179             break;
180
181         case 8:
182             printf("Exiting program...\n");
183             exit(0);
184
185         default:
186             printf("Invalid choice. Try again.\n");
187         }
```

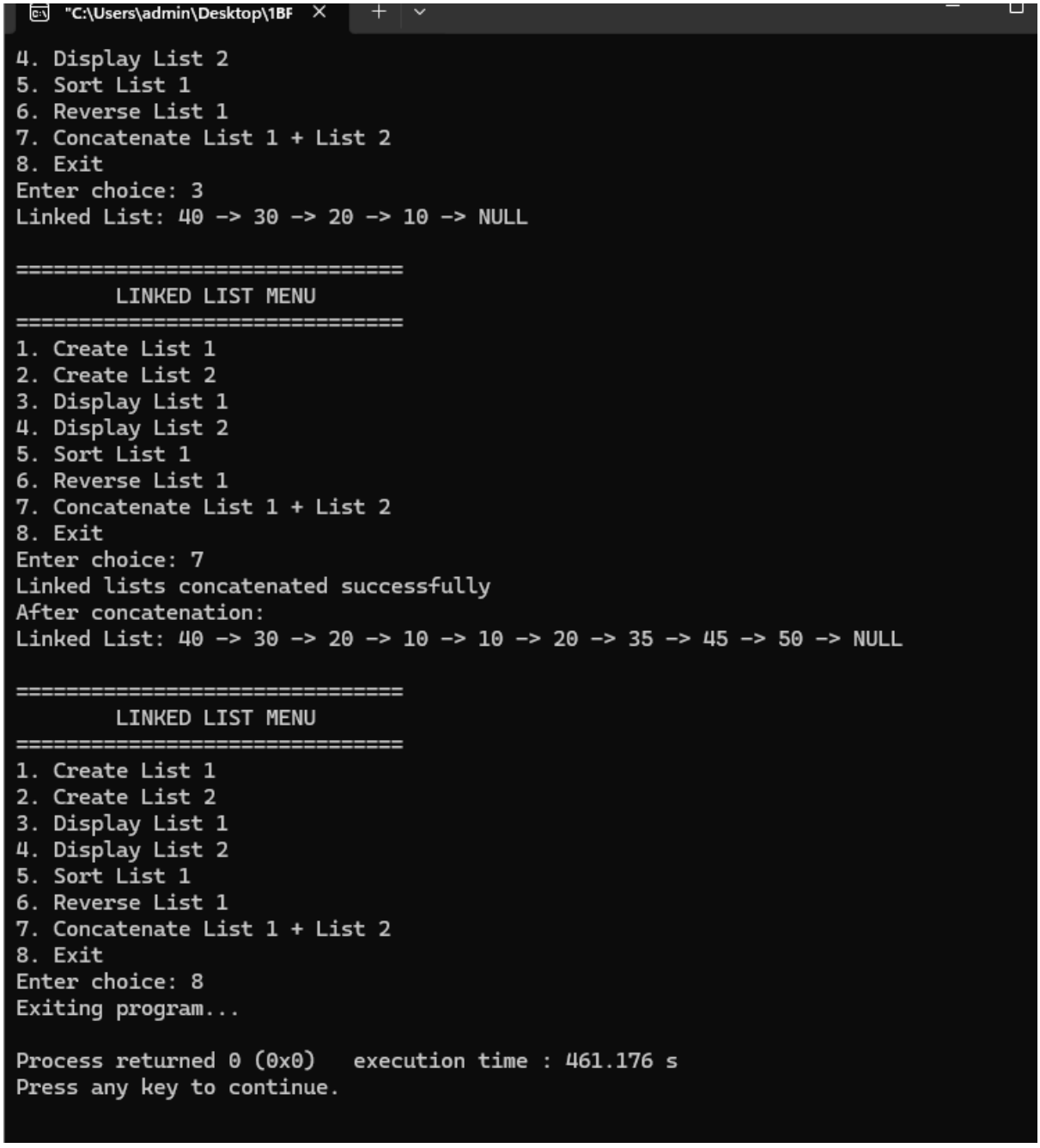
Logs & others

```
            break;

        case 6:
            head1 = reverseLinkedList(head1);
            break;

        case 7:
            head1 = concatenateLinkedList(head1, head2);
            printf("After concatenation:\n");
            displayList(head1);
            break;

        case 8:
            printf("Exiting program...\n");
            exit(0);

        default:
            printf("Invalid choice. Try again.\n");
        }
    }

    return 0;
}
```

```
==============================
        LINKED LIST MENU
==============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 1
Enter number of nodes for List 1: 4
Enter data for node 1: 40
Enter data for node 2: 20
Enter data for node 3: 30
Enter data for node 4: 10
Linked list created successfully

==============================
        LINKED LIST MENU
==============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 2
Enter number of nodes for List 2: 5
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 35
Enter data for node 4: 45
Enter data for node 5: 50
Linked list created successfully

==============================
```

```
==============================
        LINKED LIST MENU
==============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 3
Linked List: 40 -> 20 -> 30 -> 10 -> NULL

==============================
        LINKED LIST MENU
==============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 4
Linked List: 10 -> 20 -> 35 -> 45 -> 50 -> NULL

==============================
        LINKED LIST MENU
==============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 5
```

```
8. Exit
Enter choice: 5
Linked list sorted successfully

==============================
        LINKED LIST MENU
==============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 3
Linked List: 10 -> 20 -> 30 -> 40 -> NULL

==============================
        LINKED LIST MENU
==============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
7. Concatenate List 1 + List 2
8. Exit
Enter choice: 6
Linked list reversed successfully

==============================
        LINKED LIST MENU
==============================
1. Create List 1
2. Create List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Reverse List 1
```



Lab 6b

```
#include <stdio.h>

#include <stdlib.h>


struct Node {

   int data;

   struct Node *next;

};
```

```
// Stack pointers
struct Node *top = NULL;

// Queue pointers
struct Node *front = NULL;
struct Node *rear = NULL;



// Function to create a new node
struct Node* createNode(int value) {
    struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        printf("Memory allocation failed!\n");
        exit(0);
    }
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}

////////////////////////////////////////////////////////
//                STACK OPERATIONS                //
////////////////////////////////////////////////////////

// PUSH
void push(int value) {
    struct Node *newNode = createNode(value);
    newNode->next = top;
    top = newNode;
    printf("Pushed: %d\n", value);
```

```
}

// POP
void pop() {
   if (top == NULL) {
      printf("Stack Underflow!\n");
      return;
   }
   struct Node *temp = top;
   printf("%d popped from the stack\n", temp->data);
   top = top->next;
   free(temp);
}

// DISPLAY STACK
void displayStack() {
   struct Node *temp = top;
   if (temp== NULL) {
      printf("Stack is Empty!\n");
      return;
   }

   printf("Stack (top to bottom) elements are: ");
   while (temp!= NULL) {
      printf("%d ", temp->data);
      temp=temp->next;
   }
   printf("\n");
}

////////////////////////////////////////////////////////////
```

```
//                QUEUE OPERATIONS                //
////////////////////////////////////////////////////////

// ENQUEUE
void enqueue(int value) {
   struct Node *newNode = createNode(value);

   if (front == NULL) {
      front = rear = newNode;
   } else {
      rear->next = newNode;
      rear = newNode;
   }

   printf("%d enqueued to the queue\n", value);
}

// DEQUEUE
void dequeue() {
   if (front == NULL) {
      printf("Queue is empty\n");
      return;
   }

   struct Node *temp = front;
   printf("%d dequeued from queue\n", temp->data);

   front = front->next;
   if (front == NULL)
      rear = NULL;
```

```
    free(temp);
}

// DISPLAY QUEUE
void displayQueue() {
    struct Node *temp = front;
    if (temp == NULL) {
        printf("Queue is Empty!\n");
        return;
    }

    printf("Queue (front to rear) elements are: ");
    while (temp!= NULL) {
        printf("%d ", temp->data);
        temp=temp->next;
    }
    printf("\n");
}

////////////////////////////////////////////////////////////

int main() {
    int choice, value, ch;

    while (1) {
        printf("\n--- Singly Linked List Simulation ---\n");
        printf("1. Stack Operations\n");
        printf("2. Queue Operations\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
switch (choice) {

   // ---------------- STACK MENU ----------------
   case 1:
      while (1) {
         printf("\n--- Stack Menu ---\n");
         printf("1. Push\n");
         printf("2. Pop\n");
         printf("3. Display Stack\n");
         printf("4. Back to Main Menu\n");
         printf("Enter your choice: ");
         scanf("%d", &ch);

         switch (ch) {
            case 1:
               printf("Enter value to push: ");
               scanf("%d", &value);
               push(value);
               break;

            case 2:
               pop();
               break;

            case 3:
               displayStack();
               break;

            case 4:
               goto main_menu;
```

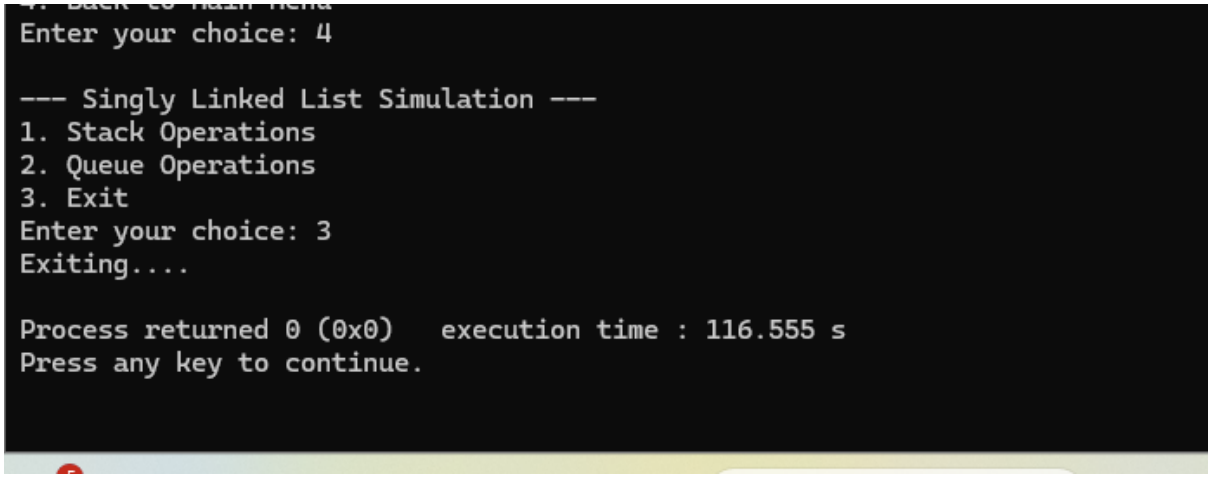
```
            default:
                printf("Invalid Choice!\n");
        }
    }
    break;

// ---------- QUEUE MENU ----------
case 2:
    while (1) {
        printf("\n--- Queue Menu ---\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display Queue\n");
        printf("4. Back to Main Menu\n");
        printf("Enter your choice: ");
        scanf("%d", &ch);

        switch (ch) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;

            case 2:
                dequeue();
                break;

            case 3:
                displayQueue();
                break;
```

```
                case 4:
                    goto main_menu;
                default:
                    printf("Invalid Choice!\n");
                }
            }
            break;

        // ---------- EXIT ----------
        case 3:
            printf("Exiting....\n");
            exit(0);

        default:
            printf("Invalid choice!\n");
    }
  main_menu: ;
  }
  return 0;
}
```

```
--- Singly Linked List Simulation ---
1. Stack Operations
2. Queue Operations
3. Exit
Enter your choice: 1

--- Stack Menu ---
1. Push
2. Pop
3. Display Stack
4. Back to Main Menu
Enter your choice: 1
Enter value to push: 10
Pushed: 10

--- Stack Menu ---
1. Push
2. Pop
3. Display Stack
4. Back to Main Menu
Enter your choice: 1
Enter value to push: 20
Pushed: 20

--- Stack Menu ---
1. Push
2. Pop
3. Display Stack
4. Back to Main Menu
Enter your choice: 1
Enter value to push: 30
Pushed: 30

--- Stack Menu ---
1. Push
2. Pop
3. Display Stack
4. Back to Main Menu
Enter your choice: 3
Stack (top to bottom) elements are: 30 20 10
```

```
--- Stack Menu ---
1. Push
2. Pop
3. Display Stack
4. Back to Main Menu
Enter your choice: 2
30 popped from the stack

--- Stack Menu ---
1. Push
2. Pop
3. Display Stack
4. Back to Main Menu
Enter your choice: 4

--- Singly Linked List Simulation ---
1. Stack Operations
2. Queue Operations
3. Exit
Enter your choice: 4
Invalid choice!

--- Singly Linked List Simulation ---
1. Stack Operations
2. Queue Operations
3. Exit
Enter your choice: 2

--- Queue Menu ---
1. Enqueue
2. Dequeue
3. Display Queue
4. Back to Main Menu
Enter your choice: 1
Enter value to enqueue: 5
5 enqueued to the queue

--- Queue Menu ---
1. Enqueue
2. Dequeue
3. Display Queue
```

```
--- Queue Menu ---
1. Enqueue
2. Dequeue
3. Display Queue
4. Back to Main Menu
Enter your choice: 1
Enter value to enqueue: 4
4 enqueued to the queue

--- Queue Menu ---
1. Enqueue
2. Dequeue
3. Display Queue
4. Back to Main Menu
Enter your choice: 1
Enter value to enqueue: 6
6 enqueued to the queue

--- Queue Menu ---
1. Enqueue
2. Dequeue
3. Display Queue
4. Back to Main Menu
Enter your choice: 3
Queue (front to rear) elements are: 5 4 6

--- Queue Menu ---
1. Enqueue
2. Dequeue
3. Display Queue
4. Back to Main Menu
Enter your choice: 2
5 dequeued from queue

--- Queue Menu ---
1. Enqueue
2. Dequeue
3. Display Queue
4. Back to Main Menu
Enter your choice: 4
```


4. Back to Main Menu
Enter your choice: 4
--- Singly Linked List Simulation ---
1. Stack Operations
2. Queue Operations
3. Exit
Enter your choice: 3
Exiting....
Process returned 0 (0x0) execution time : 116.555 s
Press any key to continue.