

```
--- 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
10 pushed onto the stack.

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

--- Stack Menu ---
1. Push
2. Pop
3. Display Stack
4. Back to Main Menu
Enter your choice: 3
Stack (Top to Bottom): 10

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

```
--- Queue Menu ---
1. Enqueue
2. Dequeue
3. Display Queue
4. Back to Main Menu
Enter your choice: 1
Enter value to enqueue: 190
190 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: 20
20 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): 190 20

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

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

Process returned 0 (0x0)   execution time : 48.086 s
Press any key to continue.
```

```

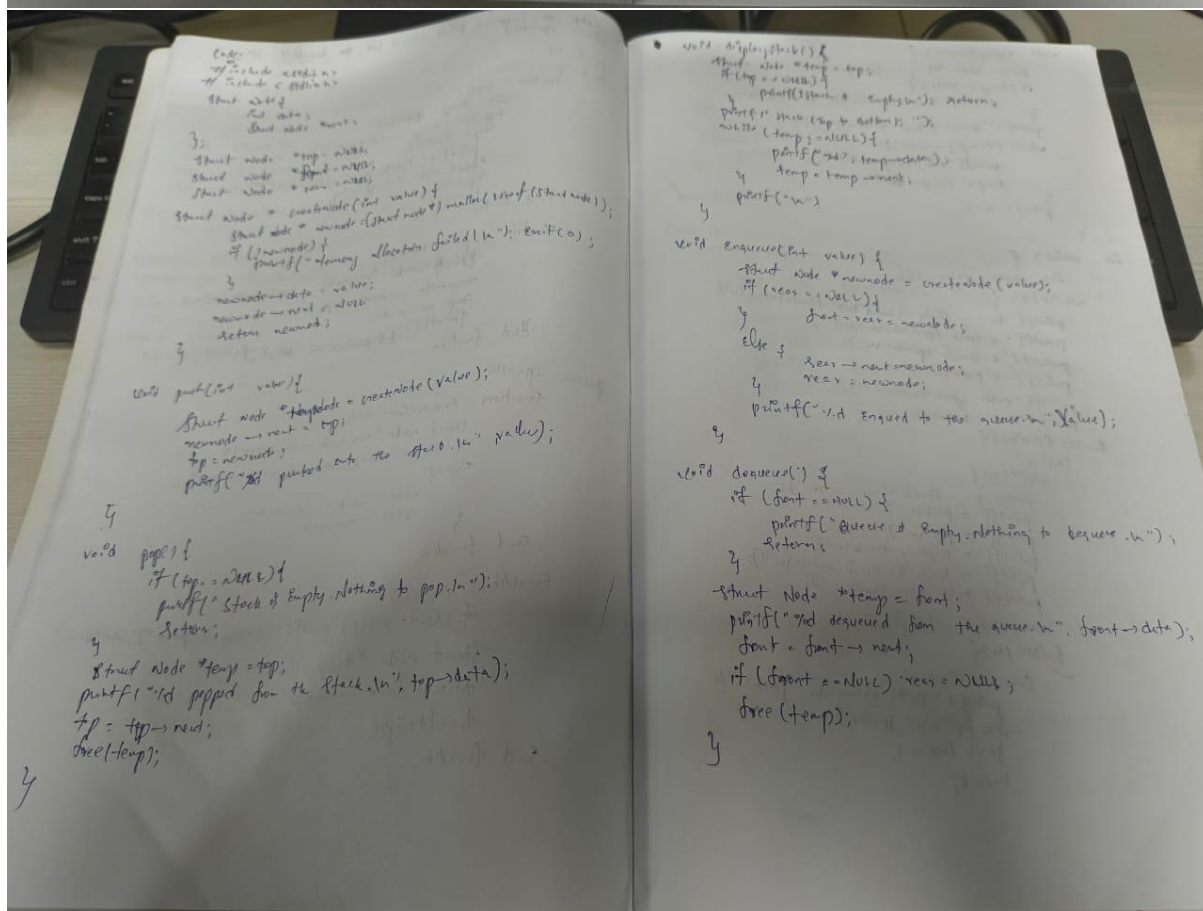
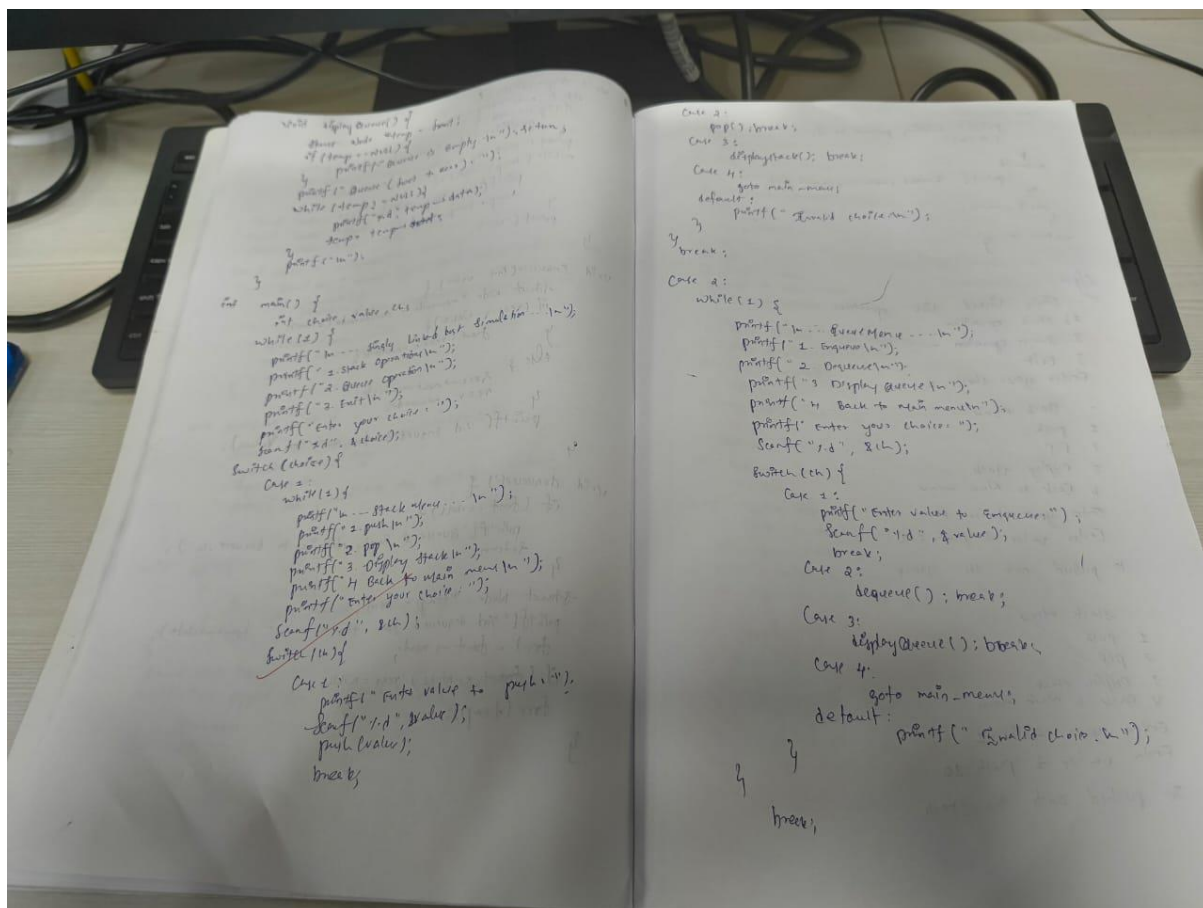
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node {
5      int data;
6      struct Node *next;
7  };
8
9  struct Node *top = NULL;
10 struct Node *rear = NULL;
11 struct Node *front = NULL;
12
13 struct Node* createNode(int value) {
14     struct Node *newNode = (struct Node*)malloc(sizeof(struct Node));
15     if (!newNode) {
16         printf("Memory allocation failed!\n");
17         return;
18     }
19     newNode->data = value;
20     newNode->next = NULL;
21     return newNode;
22 }
23
24 void push(int value) {
25     struct Node *newNode = createNode(value);
26     newNode->next = top;
27     top = newNode;
28     printf("%d pushed onto the stack.\n", value);
29 }
30
31 void pop() {
32     if (top == NULL) {
33         printf("Stack is empty. Nothing to pop. \n");
34         return;
35     }
36     struct Node *temp = top;
37     printf("%d popped from the stack.\n", top->data);
38     top = top->next;
39     free(temp);
40 }
41
42 void displayStack() {
43     struct Node *temp = top;
44     if (temp == NULL) {
45         printf("Stack is empty.\n");
46         return;
47     }
48     printf("Stack (Top to Bottom): ");
49     while (temp != NULL) {
50         printf("%d ", temp->data);
51         temp = temp->next;
52     }
53     printf("\n");
54 }
55
56 void enqueue(int value) {
57     struct Node *newNode = createNode(value);
58     if (rear == NULL) {
59         front = rear = newNode;
60     }
61     else {
62         rear->next = newNode;
63         rear = newNode;
64     }
65     printf("%d enqueued to the queue.\n", value);
66 }
67
68
69
70 void dequeue() {
71     if (front == NULL) {
72         printf("Queue is empty. Nothing to dequeue. \n");
73         return;
74     }
75     struct Node *temp = front;
76     printf("%d dequeued from the queue. \n", front->data);
77     front = front->next;
78
79     if (front == NULL) {
80         rear = NULL;
81     }
82     free(temp);
83 }
84
85
86 void displayQueue() {
87     struct Node *temp = front;
88     if (temp == NULL) {
89         printf("Queue is empty.\n");
90         return;
91     }
92     printf("Queue (Front to Rear): ");
93     while (temp != NULL) {
94         printf("%d ", temp->data);
95         temp = temp->next;
96     }
97     printf("\n");
98 }
99
100

```

```

101 int main() {
102     int choice, value, ch;
103
104     while (1) {
105         printf("\n --- singly Linked List Simulation --- \n");
106         printf("1. Stack Operations\n");
107         printf("2. Queue Operations\n");
108         printf("3. Exit\n");
109         printf("Enter your choice: ");
110         scanf("%d", &choice);
111
112         switch (choice) {
113             case 1:
114
115                 while (1) {
116                     printf("\n --- Stack Menu --- \n");
117                     printf("1. Push\n");
118                     printf("2. Pop\n");
119                     printf("3. Display Stack\n");
120                     printf("4. Back to Main Menu\n");
121                     printf("Enter your choice: ");
122                     scanf("%d", &ch);
123
124                     switch (ch) {
125                         case 1:
126                             printf("Enter value to push: ");
127                             scanf("%d", &value);
128                             push(value);
129                             break;
130
131                         case 2:
132                             pop();
133                             break;
134                         case 3:
135                             displayStack();
136                             break;
137                         case 4:
138                             goto main_menu;
139                         default:
140                             printf("Invalid choice.\n");
141                     }
142                 }
143             }
144         break;
145     case 2:
146
147         while (1) {
148             printf("\n --- Queue Menu --- \n");
149             printf("1. Enqueue\n");
150             printf("2. Dequeue\n");
151             printf("3. Display Queue\n");
152             printf("4. Back to Main Menu\n");
153             printf("Enter your choice: ");
154             scanf("%d", &ch);
155
156             switch (ch) {
157                 case 1:
158                     printf("Enter value to enqueue: ");
159                     scanf("%d", &value);
160                     enqueue(value);
161                     break;
162                 case 2:
163                     dequeue();
164                     break;
165                 case 3:
166                     displayQueue();
167                     break;
168                 case 4:
169                     goto main_menu;
170                 default:
171                     printf("Invalid choice.\n");
172             }
173         }
174     }
175     break;
176 }
177 case 3:
178     printf("Exiting program.\n");
179     exit(0);
180 default:
181     printf("Invalid choice. Try again.\n");
182 }
183 main_menu: ;
184 }
185 return 0;
186 }
187 }
188 }

```



Q/A

1. Slightly twisted list operations -

2. Short operations

2. Merge operation

3. Exit

Enter your choice: 2

10 pushed out the stalk

1. push
2. pop
3. Display stack
4. Back to main menu

var lue to push: 20

☆

- Stack frame -
- 1. push
- 2. pop
- 3. Display stack
- 4. Back to main menu
- Enter your choice :

Enter your choice: \_\_\_\_\_

1. Enqueue
2. Dequeue
3. Display Queue
4. Back to Main menu

2. Des veld

4. Back to Main menu

Enter your choice.

10 dequeued from the queue

Engraving

