

STACK AND QUEUE OPERATION:

```
#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;  
}
```

OUTPUT:

```
C:\Users\student\Desktop\chu x + v

--- 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: 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
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

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

Process returned 0 (0x0)   execution time : 131.570 s
Press any key to continue.
```

OBSERVATION:

Q4.10

Prg-6b

Date: / /
Page: /

b. WAP to implement single link list to simulate stack & queue operations.

Ans: Pseudocode:-

CreateNode(value)

→ Create a new node
Set node.data = value
Set node.next = NULL
Return node.

Stack Operations:-

Push(value)

→ newNode = CreateNode(value)
newNode.next = top
top = newNode
Display "value pushed to stack"

pop()

→ If top is Null
Display "stack is empty"
Return

temp = top

Display temp.data + "popped from stack"

top = top.next

Delete temp.

Display stack()

→ If top is Null

Display "Stack is empty"
Return

temp = top

Display "Stack (top to Bottom);"

while temp is not NULL

Display temp.data

temp = temp.next

Queue Operations

Enqueue (uvalue)

→ newNode = createNode (uvalue)

if rear == NULL

front = rear = newNode

else

rear → next = newNode

rear = newNode

display ("enqueued element");

dequeue()

→ If front == NULL
printf ("empty")
return

temp = front

printf ("dequeued element")

front = front → next

if front == NULL

rear = NULL

free (temp)

display()

→ If front == NULL

printf ("empty")

return

temp = front

print ("queue element")

while (temp != NULL)

printf (temp → data)

temp = temp → next

```
#include <stdio.h>
#include <stdlib.h>
```

```
struct node {
    int data;
    struct node * next;
};
```

```
struct node * top = NULL;
struct node * front = NULL;
struct node * rear = NULL;
```

```
struct node * createNode(str int value) {
    struct node * newNode = (struct node *) malloc
                             (size of (struct node));
    if (!newNode) {
        printf("Memory allocation failed\n");
        return;
    }
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}
```

Stack:-

```
void push(int value) {
    struct node * newNode = createNode(value);
    newNode->next = top;
    top = newNode;
    printf("%d pushed into the stack\n",
           value);
}
```

```
void pop() {
    if (top == NULL) {
        printf("Stack is empty\n");
        return;
    }
    struct node * temp = top;
    printf("%d popped from the stack\n",
        top->data);
    top = top->next;
    free(temp);
}
```

```
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

```
void enqueue(int value) {
    struct node * newNode = createNode(value);
    if (rear == NULL) {
        front = rear = newNode;
    }
}
```

```

else {
    rear->next = newNode;
    newNode = rear;
}
printf(" %d enqueued to the queue\n",
      value);
}

```

```

void dequeue() {
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }
    struct node *temp = front;
    printf(" %d dequeued from the queue\n",
      front->data);
    front = front->next;

    if (front == NULL)
        rear = NULL;

    free(temp);
}

```

```

void displayQueue() {
    struct node *temp = front;
    if (temp == NULL) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements (front to rear) 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) {

~~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:

```
1: printf ("Enter value to push : ");
```

Continued psg 6.

```
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;
```

```
case 2:
    while (1) {
        printf ("\n --- Queue Menu --- \n");
        printf ("1. Enqueue\n");
        printf ("2. Dequeue\n");
        printf ("3. DisplayQueue\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;
```

```
case 3:
```

```
    printf ("Exiting.... \n");  
    exit (0);
```

```
default:
```

```
    printf ("Invalid choice \n");
```

```
}
```

```
main-menu;
```

```

    }
    return 0;
}

```

Output :-

----- 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: 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
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 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...

By
 1/1/20
 Dev

