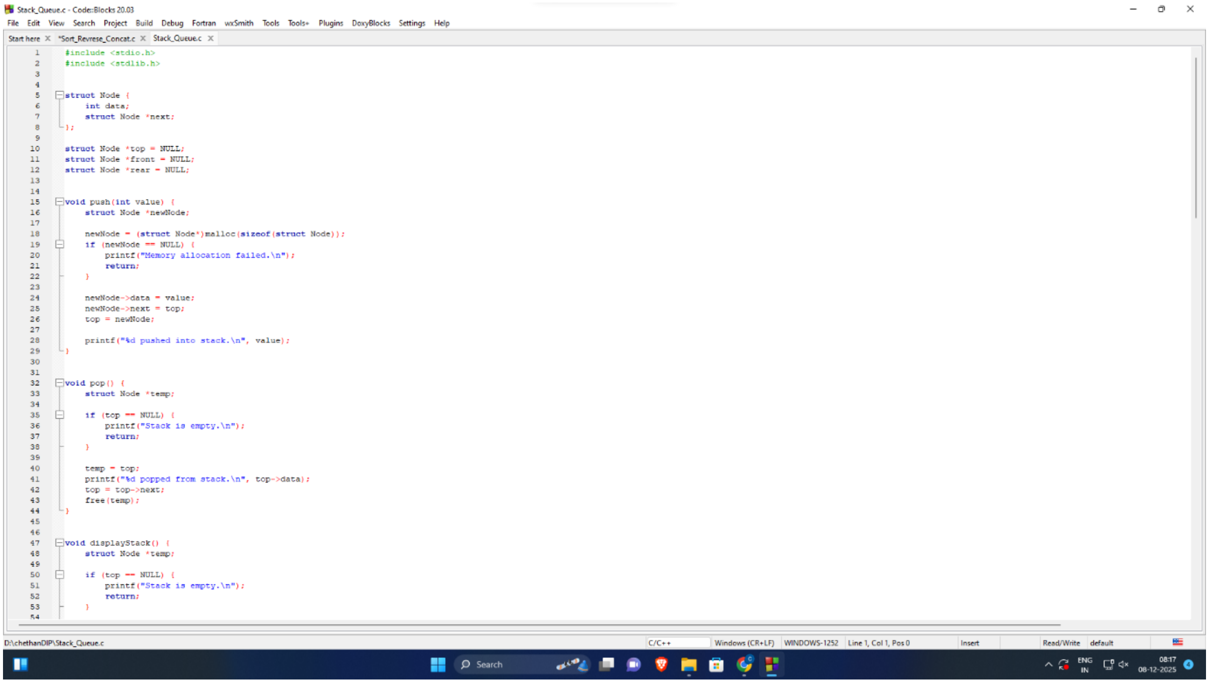
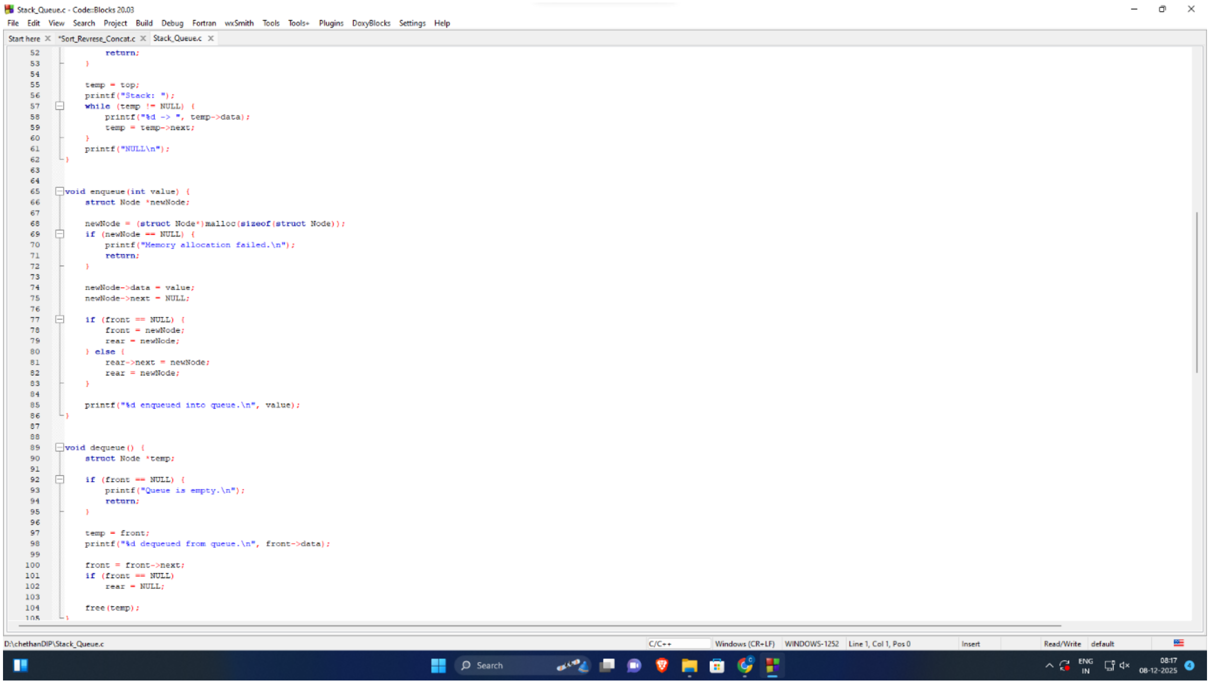
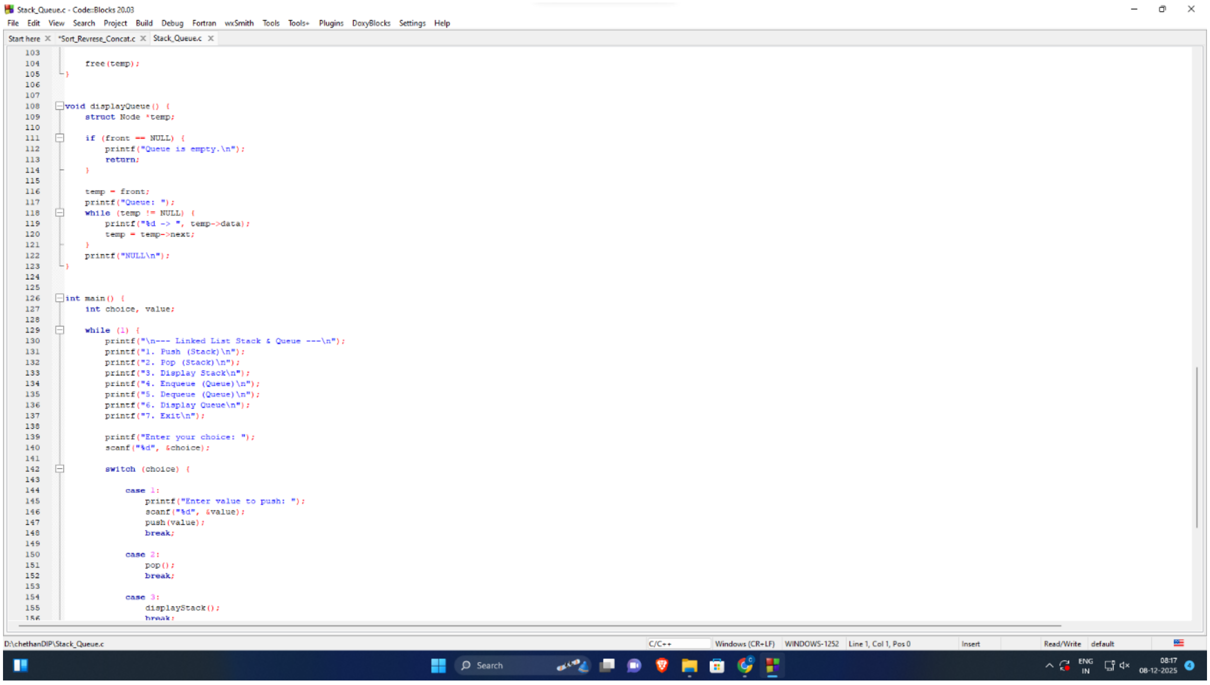

Stack_Queue.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
*Sort_Revrese_Concat.c
Stack_Queue.c
D:\chethanDIP\Stack_Queue.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default
Search
ENG
IN
08:17
08-12-2025

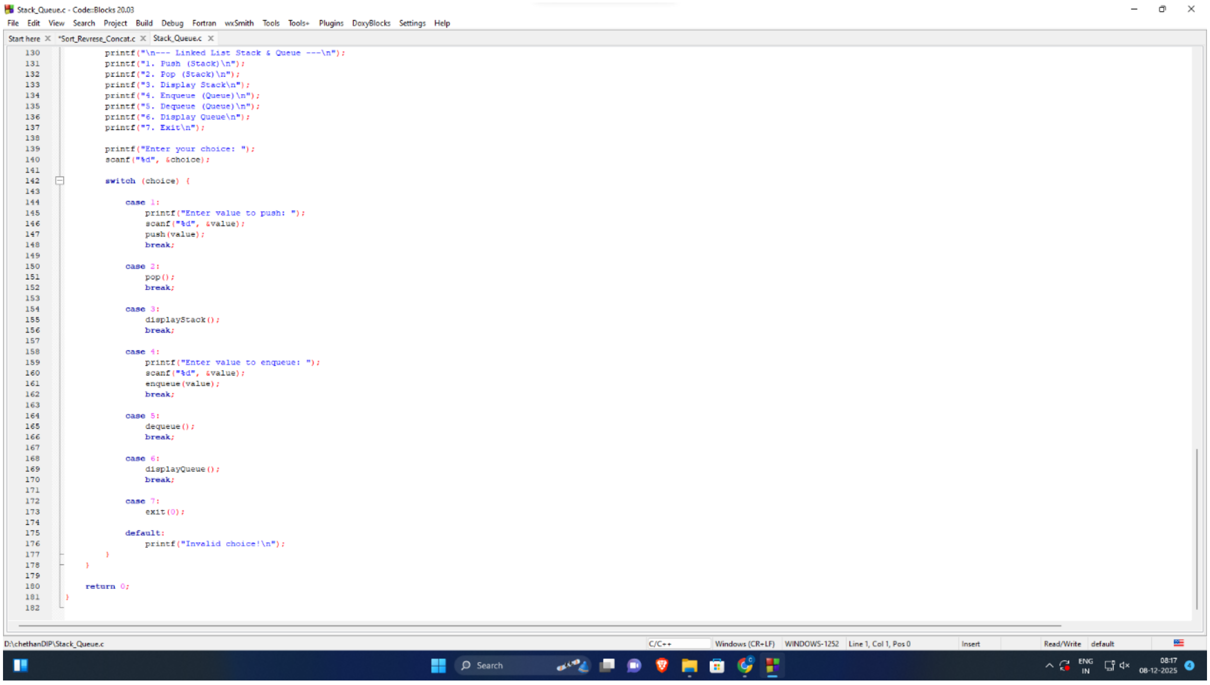
```c
#include <stdio.h>
#include <stdlib.h>


struct Node {
    int data;
    struct Node *next;
};

struct Node *top = NULL;
struct Node *front = NULL;
struct Node *rear = NULL;


void push(int value) {
    struct Node *newNode;

    newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }

    newNode->data = value;
    newNode->next = top;
    top = newNode;

    printf("%d pushed into stack.\n", value);
}


void pop() {
    struct Node *temp;

    if (top == NULL) {
        printf("Stack is empty.\n");
        return;
    }

    temp = top;
    printf("%d popped from stack.\n", top->data);
    top = top->next;
    free(temp);
}


void displayStack() {
    struct Node *temp;

    if (top == NULL) {
        printf("Stack is empty.\n");
        return;
    }

```


Stack_Queue.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
*Sort_Revrese_Concat.c
Stack_Queue.c
D:\chethanDIP\Stack_Queue.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default
Search
ENG
IN
08:17
08-12-2025

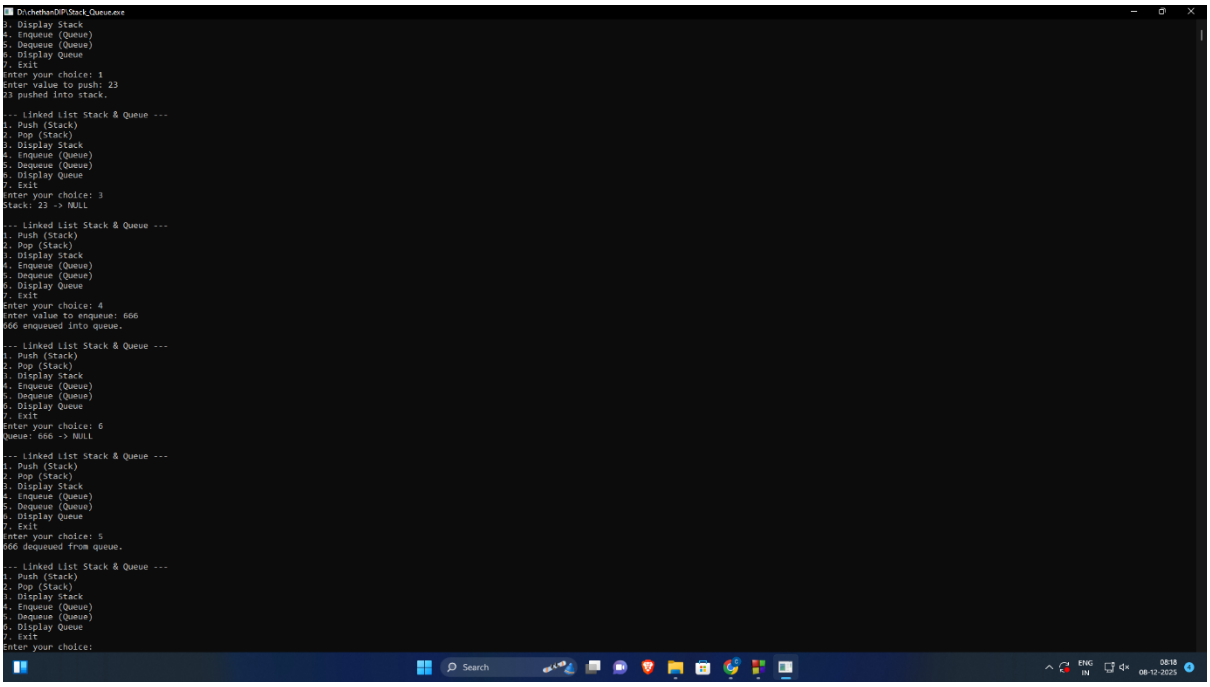
```c
        return;
    }

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


void enqueue(int value) {
    struct Node *newNode;

    newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }

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

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

    printf("%d enqueued into queue.\n", value);
}


void dequeue() {
    struct Node *temp;

    if (front == NULL) {
        printf("Queue is empty.\n");
        return;
    }

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

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

    free(temp);
}
```


Stack_Queue.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
*Sort_Revrese_Concat.c
Stack_Queue.c
D:\chethanDIP\Stack_Queue.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default
Search
ENG
IN
08:17
08-12-2025

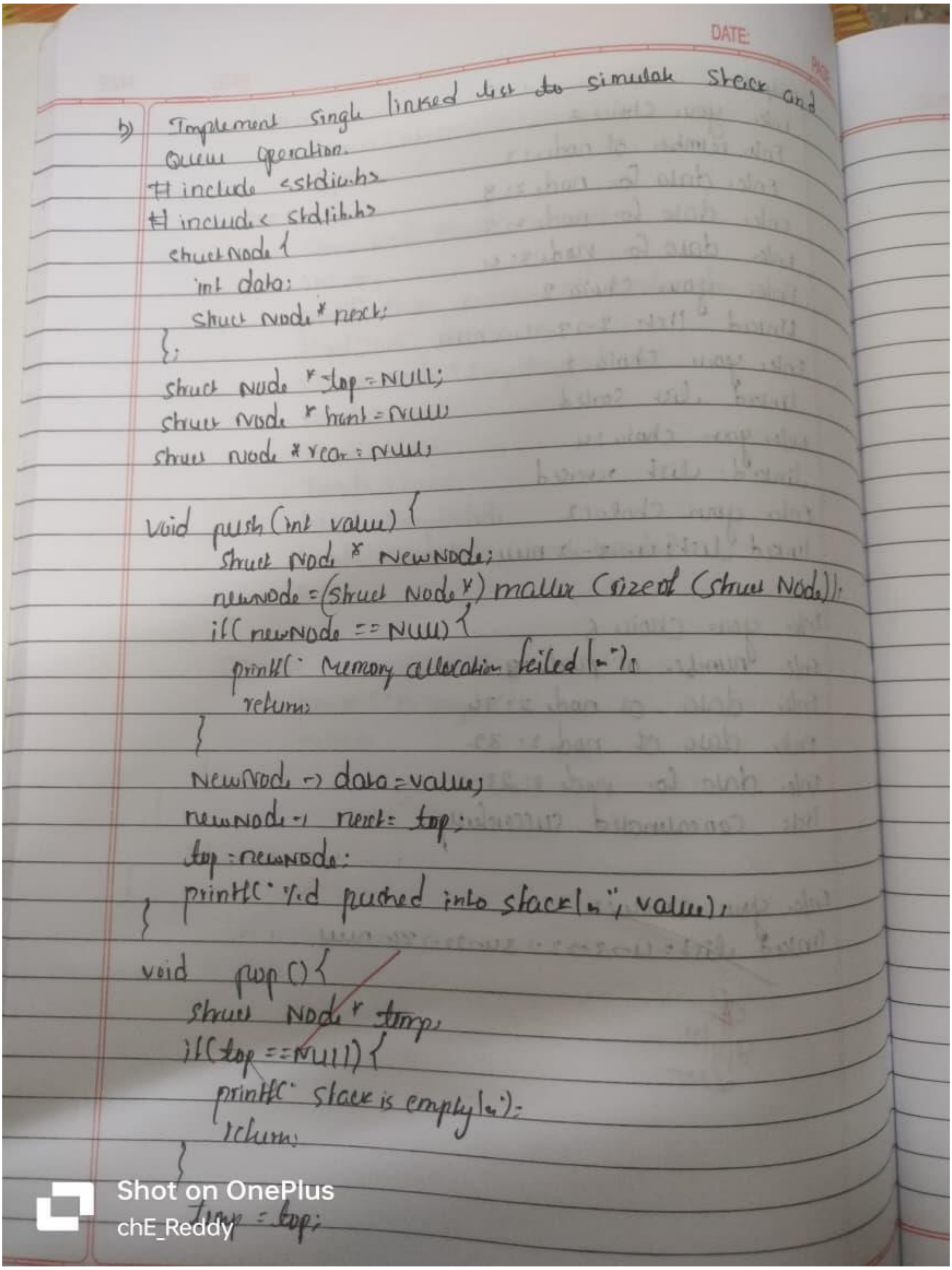
```c

    free(temp);
}


void displayQueue() {
    struct Node *temp;

    if (front == NULL) {
        printf("Queue is empty.\n");
        return;
    }

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


int main() {
    int choice, value;

    while (1) {
        printf("\n--- Linked List Stack & Queue ---\n");
        printf("1. Push (Stack)\n");
        printf("2. Pop (Stack)\n");
        printf("3. Display Stack\n");
        printf("4. Enqueue (Queue)\n");
        printf("5. Dequeue (Queue)\n");
        printf("6. Display Queue\n");
        printf("7. Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {

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

            case 2:
                pop();
                break;

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


Stack_Queue.c - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Start here
*Sort_Revrese_Concat.c
Stack_Queue.c
D:\chethanDIP\Stack_Queue.c
C/C++
Windows (CR+LF)
WINDOWS-1252
Line 1, Col 1, Pos 0
Insert
Read/Write
default
Search
ENG
IN
08:17
08-12-2025

```c
        printf("\n--- Linked List Stack & Queue ---\n");
        printf("1. Push (Stack)\n");
        printf("2. Pop (Stack)\n");
        printf("3. Display Stack\n");
        printf("4. Enqueue (Queue)\n");
        printf("5. Dequeue (Queue)\n");
        printf("6. Display Queue\n");
        printf("7. Exit\n");

        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {

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

            case 2:
                pop();
                break;

            case 3:
                displayStack();
                break;

            case 4:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;

            case 5:
                dequeue();
                break;

            case 6:
                displayQueue();
                break;

            case 7:
                exit(0);

            default:
                printf("Invalid choice!\n");
        }
    }

    return 0;
}

```

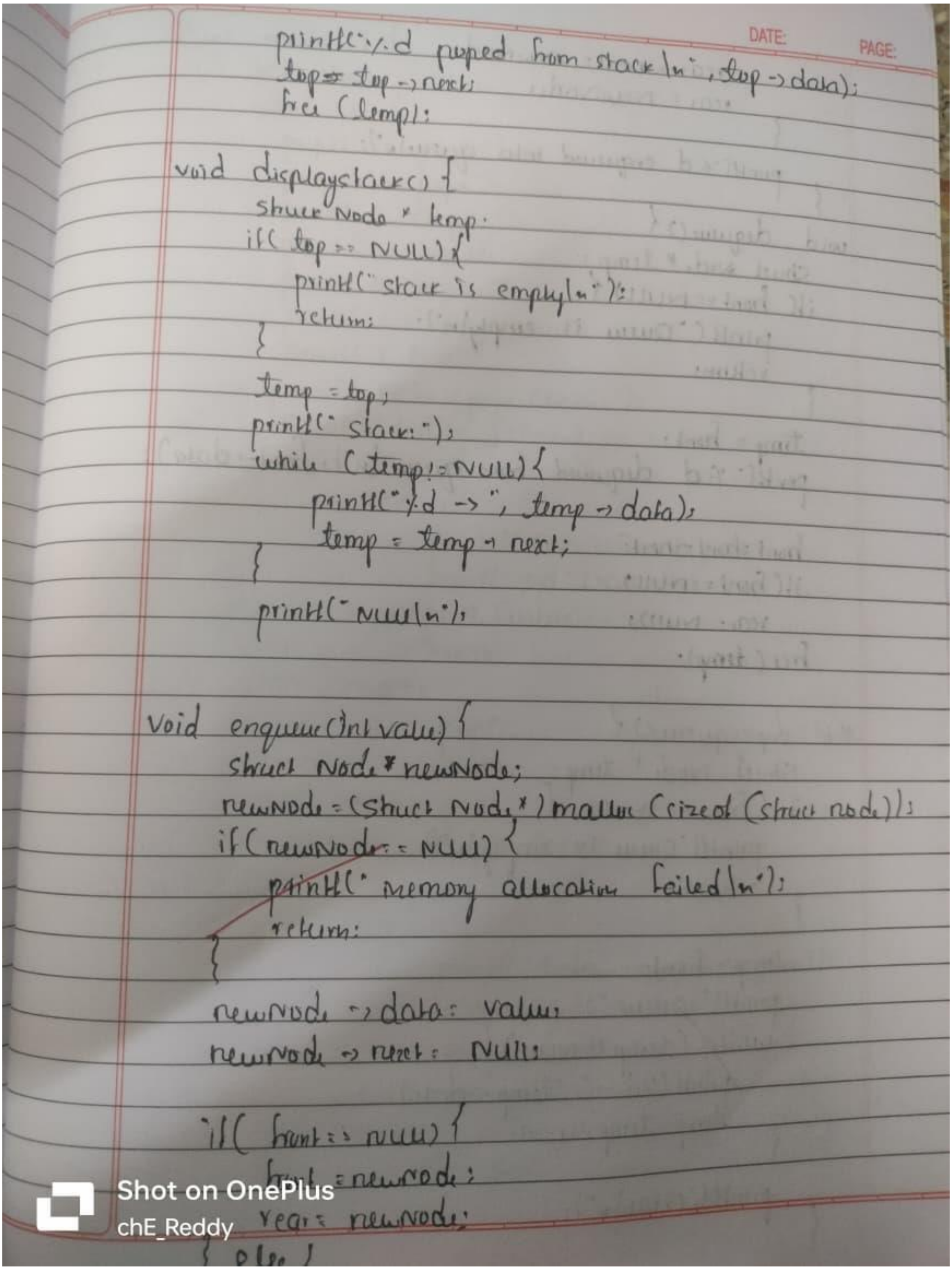


```
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 1
Enter value to push: 23
23 pushed into stack.

--- Linked List Stack & Queue ---
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 3
Stack: 23 -> NULL

--- Linked List Stack & Queue ---
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 4
Enter value to enqueue: 666
666 enqueued into queue.

--- Linked List Stack & Queue ---
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 6
Queue: 666 -> NULL

--- Linked List Stack & Queue ---
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice: 5
666 dequeued from queue.

--- Linked List Stack & Queue ---
1. Push (Stack)
2. Pop (Stack)
3. Display Stack
4. Enqueue (Queue)
5. Dequeue (Queue)
6. Display Queue
7. Exit
Enter your choice:
```

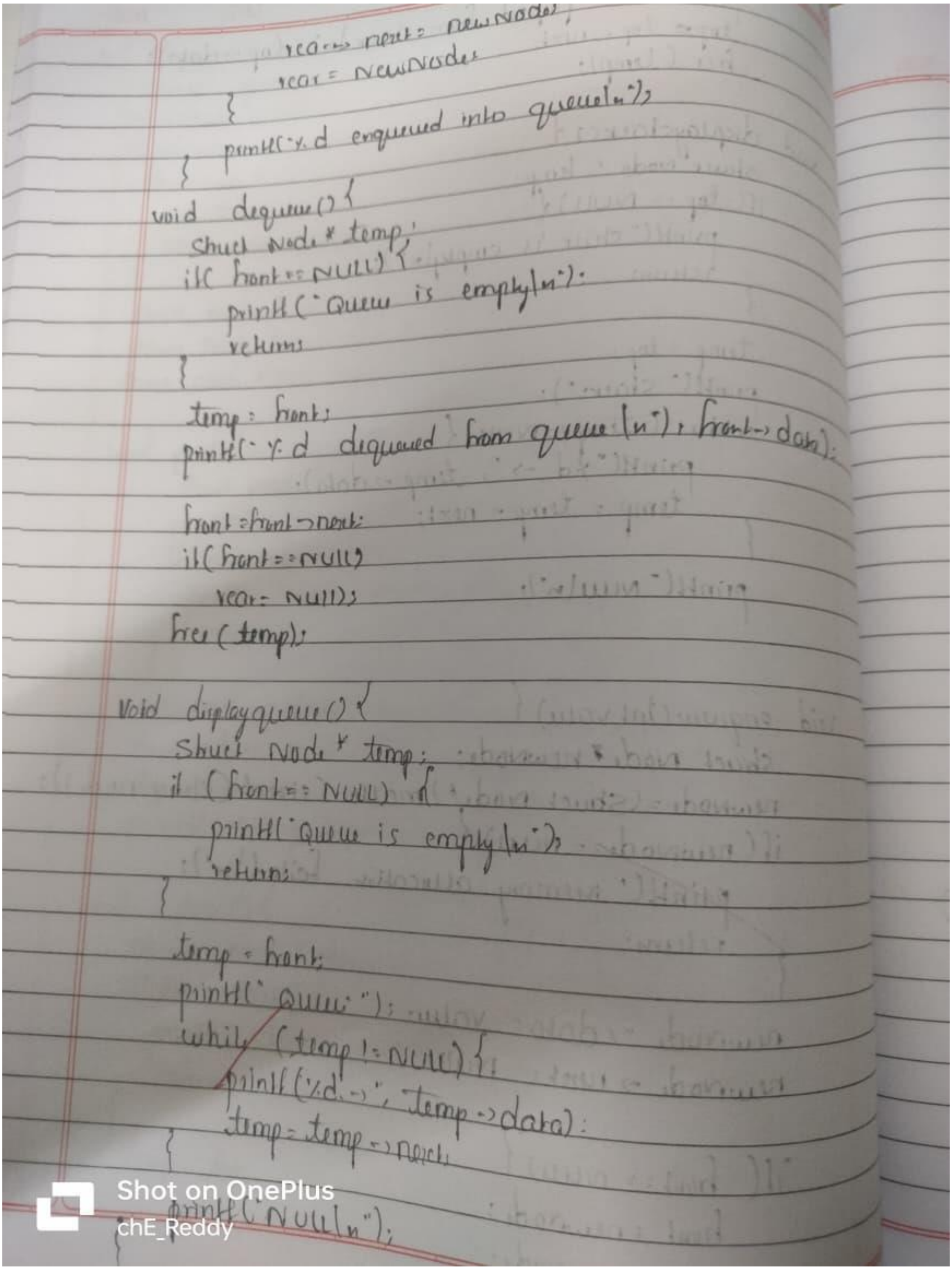


b) Implement single linked list to simulate Stack and Queue operation.

```
# 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;

void push(int value) {
    struct Node * NewNode;
    newNode =(struct Node*) malloc (sizeof (struct Node));
    if( newNode == NULL) {
      printf(" Memory allocation failed\n");
       return;
    }
   NewNode -> data=value;
   newNode -> next= top;
   top =newNode;
   printf(" %d pushed into stack\n", value);
}
void  pop(){
   struct Node * temp;
   if(top ==NULL) {
     printf(" stack is empty\n");
     return;
   }
   temp = top;
```

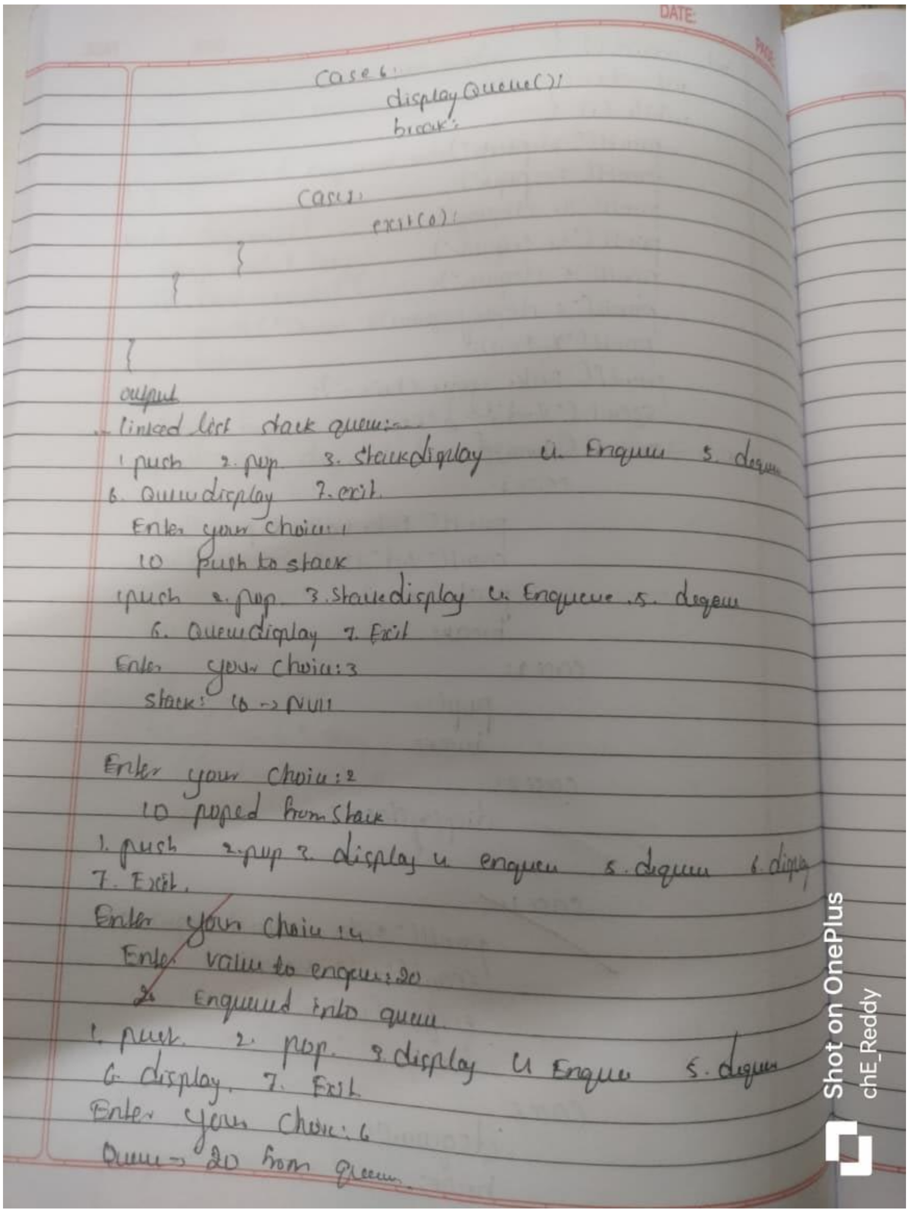


```
        printf("%d poped from stack\n", top->data);
        top = top->next;
        free (temp);

void displaystack() {
        struct Node * temp;
        if( top == NULL){
            printf(" stack is empty\n");
            return;
        }

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

void enqueue(int value) {
    struct Node *newNode;
    newNode = (struct Node*) malloc (sizeof (struct node));
    if(newNode == NULL) {
        printf(" memory allocation failed\n");
        return;
    }
    newNode -> data = value;
    newNode -> next = NULL;

    if( front == NULL) {
        front = newNode;
        rear = newNode;
    } else {
```



```
        rear->next = newNode;
        rear = NewNode;
    }
    printf("%d enqueued into queue\n");
}
void dequeue(){
    Struct Node* temp;
    if( front== NULL) {
        printf("Queue is empty\n");
        return;
    }
    temp = front;
    printf(" %d dequeued from queue \n"), front->data);

    front = front->next;
    if( front==NULL)
        rear= NULL);
    free( temp);

void displayqueue() {
    Struct Node* temp;
    if (front== NULL) {
        printf("Queue is empty\n");
        return;
    }

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

```
int main () {
    int choice, value;
    while (1) {
      printf("1. push ");
      printf("2. pop ");
      printf("3. display");
      printf ("4. enqueue ");
      printf("5. dequeue ").
      printf("6. display queue");
      printf("7. Exit");
     printf("Enter your choice: ");
     scanf ("%d", &choice);
     switch (choice) {
             case 1:
                   printf(" Enter value to push");
                   scanf("%d",&value);
                   push (value);
                   break;
           case 2:
                   pop();
                   break;
           case 3:
                   displaystack();

           case 4:
                   printf(" Enter value of enqueue");
                   scanf("%d", value);
                   enqueue (value);
                   break;
           case 5:
                   dequeue();
                   break;
```

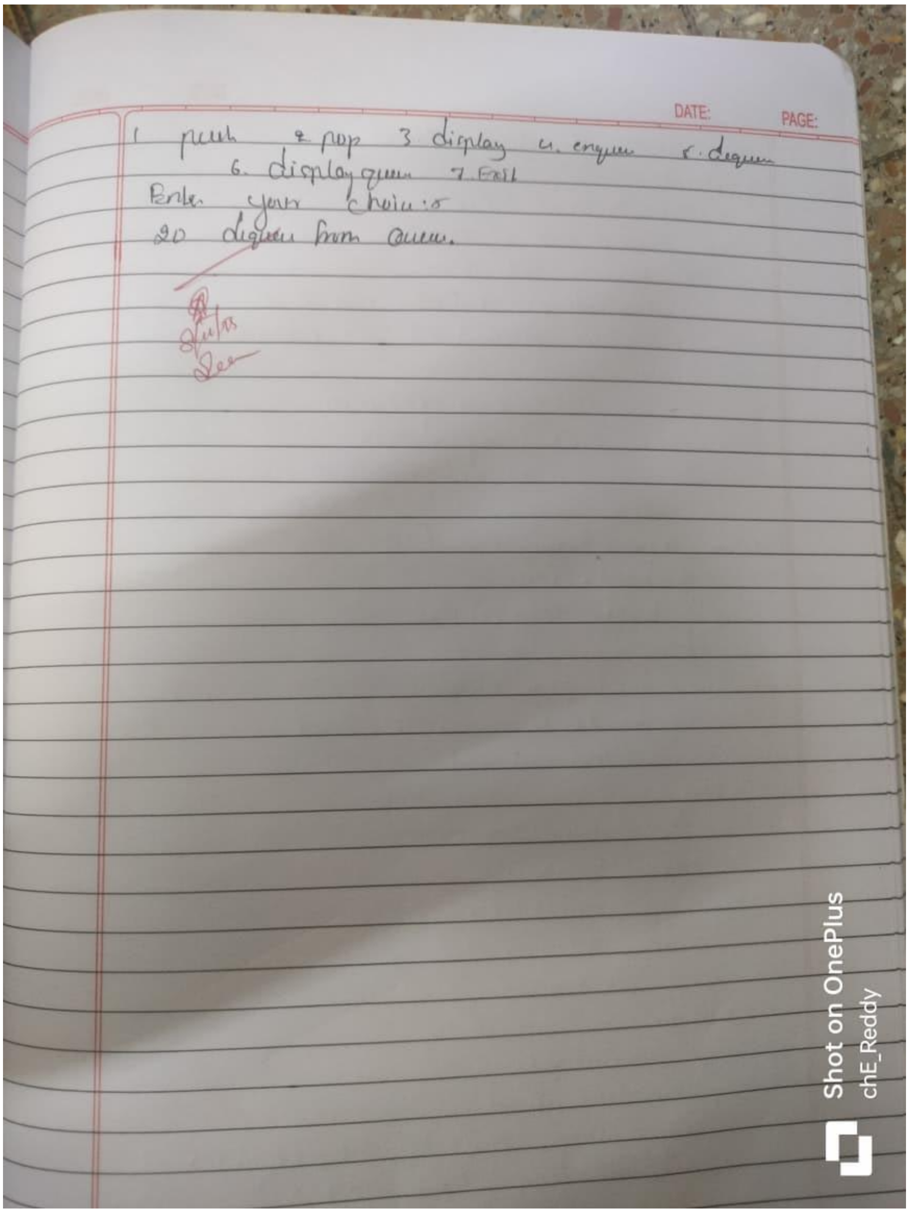


DATE:

```
            case 6:
                    display Queue();
                    break;

            case 7:
                    exit(0);
        }
    }

}
```

output

```
linked list stack queue:-
1 push   2. pop   3. stackdisplay    4. Enqueue   5. dequeu
6. Queuedisplay   7. exit.
 Enter your choice: 1
  10  push to stack
1push  2. pop  3. stackdisplay  4. Enqueue .5. dequeu
   6. Queuedisplay  7. Exit
Enter   your choice: 3
  stack: 10 -> NULL

Enter  your  choice: 2
   10  poped  from stack
1. push    2. pop 3. display 4  enqueu   5. dequeu   6. display
7. Exit.
Enter  your choice : 4
  Enter value to enqueu: 20
    20  Enqueued into queue.
1. push.   2. pop.  3. display  4 Enqueu    5. dequeu
6. display.  7. Exit
Enter  your  choice: 6
Queue -> 20 from queue.
```



DATE: PAGE:

```
1 push   2 pop  3 display  4. enqueue  5. dequeue
     6. display queue   7. Exit
Enter your choice:5
20 dequeue from queue.
```