**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**



**LAB REPORT**

**on**

**Data Structures using C Lab**

**(23CS3PCDST)**

***Submitted by***

**Subramanya J (1BM23CS343)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**Sep-2024 to Jan-2025**

**B.M.S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “Data Structures using C Lab(23CS3PCDST)”carried out by **Subramanya J(1BM23CS343),** who is bonafide student of **B.M.S.College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering inComputer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of Data Structures using C Lab(23CS3PCDST) work prescribed for the said degree.

|  |  |
| --- | --- |
| Lab faculty Incharge Name  Assistant Professor  Department of CSE, BMSCE | Dr. Kavitha Sooda  Professor &HOD  Department of CSE, BMSCE |

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 | 30.09.2024 | Stack using Array |  |
| 2 | 7.10.2024 | Infix to Postfix |  |
| 3 | 14.10.2024 | Linear Queue |  |
| 4 | 21.10.2024 | Circular Queue |  |
| 5 | 28.10.2024 | Linked List : Insertion |  |
| 6 | 11.11.2024 | Linked List : Deletion |  |
| 7 | 2.12.2024 | Linked List : Sort, Reverse, Concatenation |  |
| 8 | 16.12.2024 | Doubly Linked List |  |
| 9 | 23.12.2024 | Binary Search Tree Traversal |  |
| 10 | 23.12.2024 | Graph Traversal |  |

Github Link: https://github.com/SubramanyaJ/1BM23CS343-SubramanyaJ-data-structures

**Program 1**

Write a program to simulate the working of stack using an array with the following:

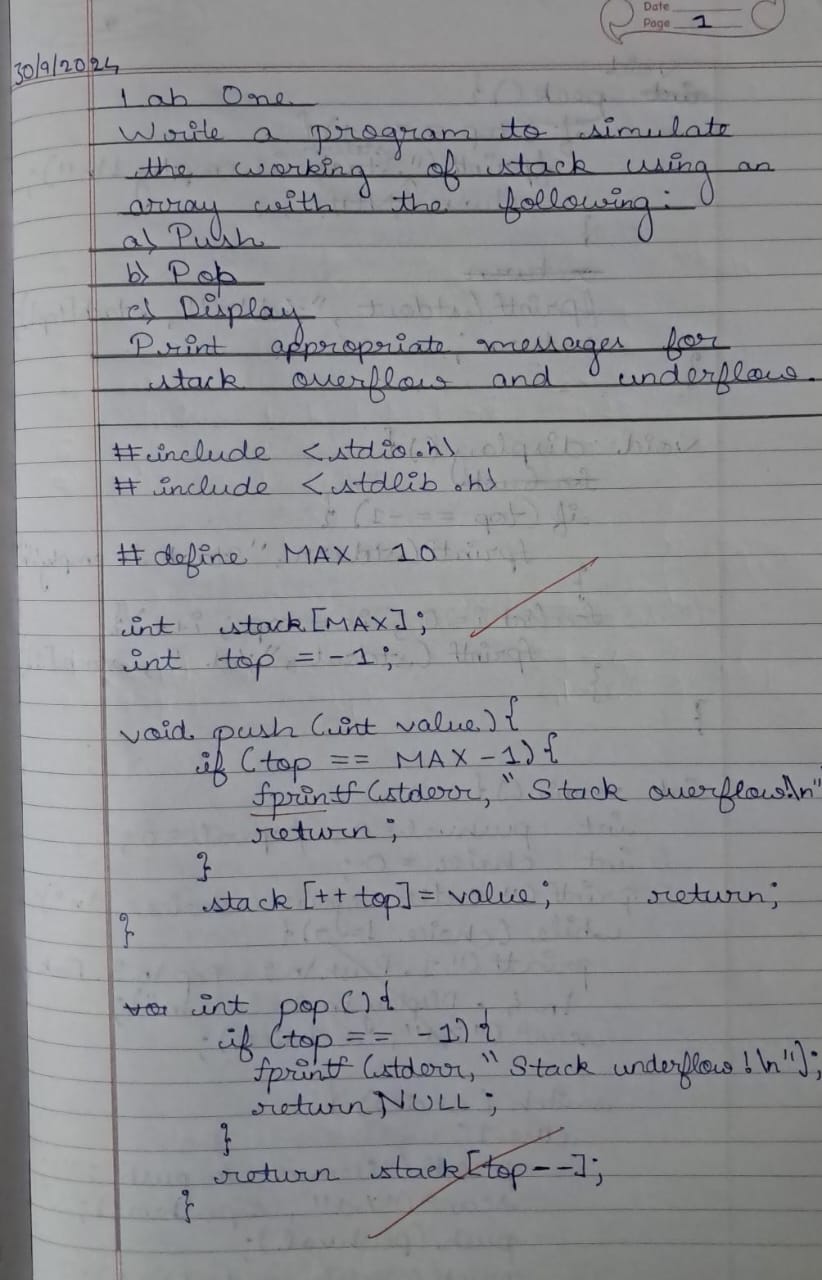
a) Push

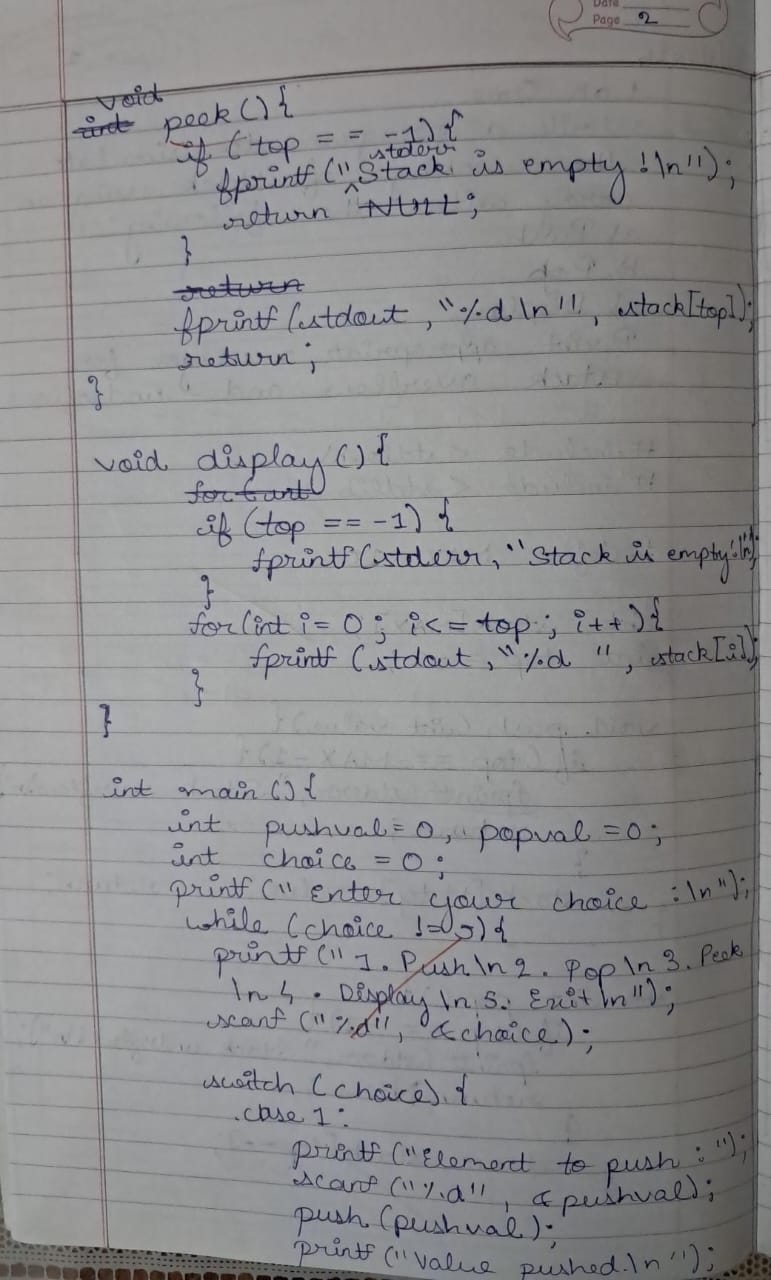
b) Pop

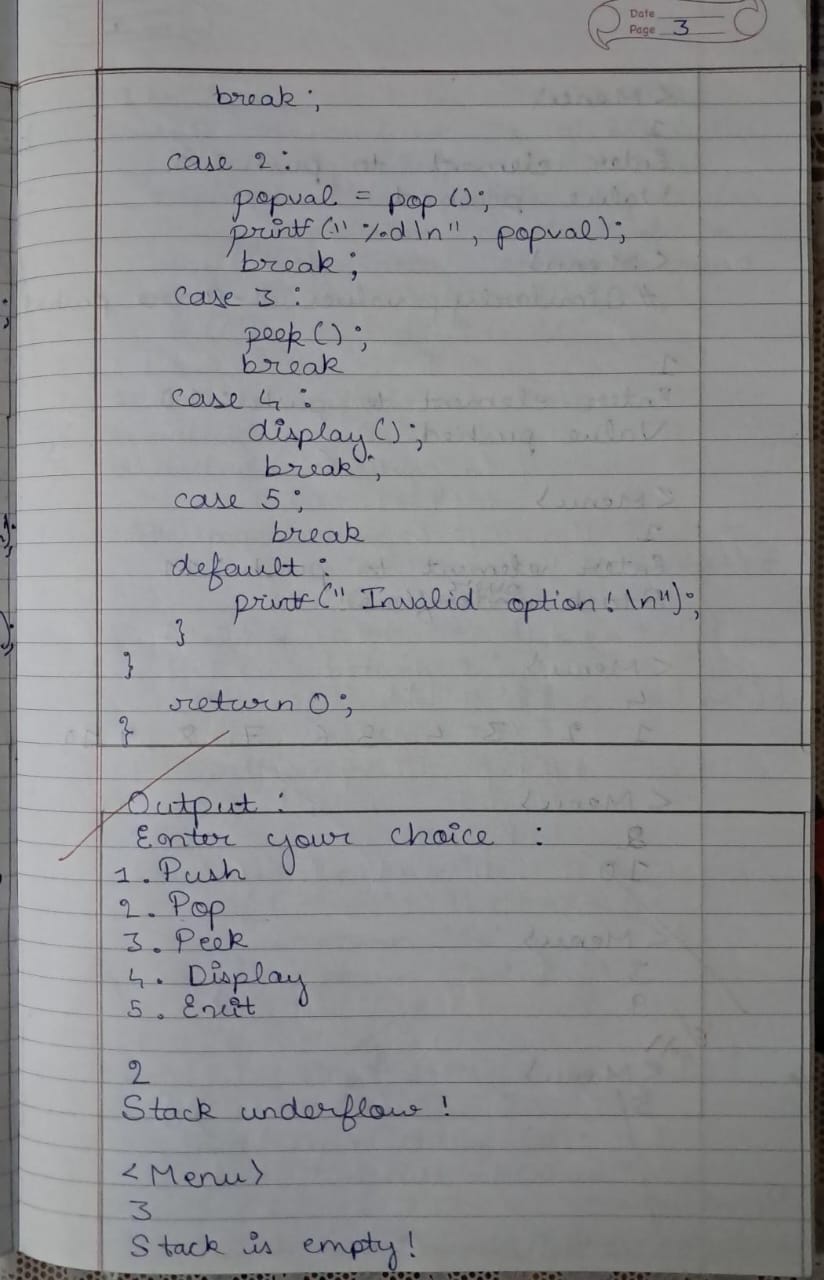
c) Display

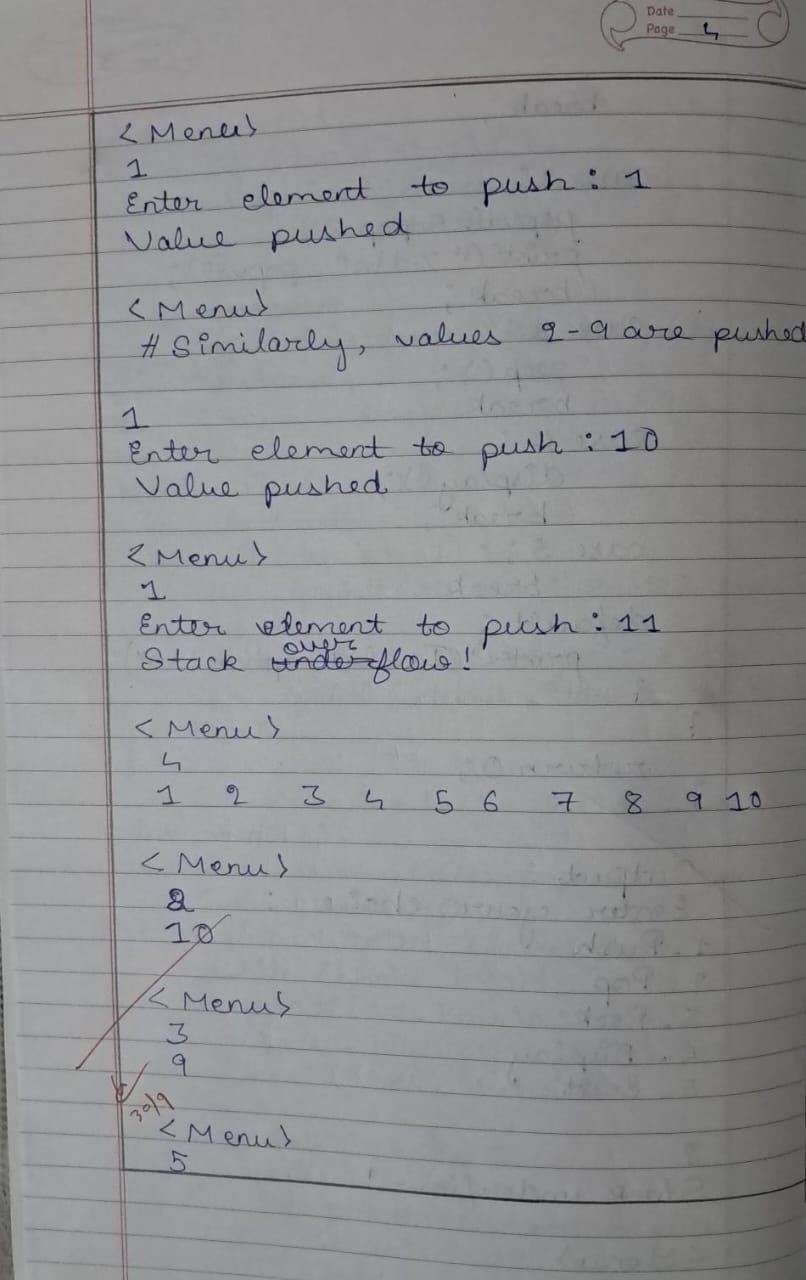
The program should print appropriate messages for stack overflow, stack underflow

Observation:









Code :

/\*

Write a program to simulate the working of stack using an

array with the following:

a) Push

b) Pop

c) Display

The program should print appropriate messages for stack

overflow, stack underflow

\*/

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX 10

int stack[MAX];

int top = -1;

void push(int value){

if(top == MAX - 1){

fprintf(stderr, "Stack overflow!\n");

return;

}

stack[++top] = value;

return;

}

int pop(){

if(top == -1){

fprintf(stderr, "Stack underflow!\n");

return 0;

}

return stack[top--];

}

void peek(){

if(top == -1){

fprintf(stderr, "Stack is empty!\n");

return;

}

fprintf(stdout, "%d\n", stack[top]);

return;

}

void display(){

if(top == -1){

fprintf(stderr, "Stack underflow!\n");

return;

}

for(int i = 0; i <= top; i++){

fprintf(stdout, "%d ", stack[i]);

}

}

int main(){

int pushval = 0, popval = 0;

int choice = 0;

printf("Enter your choice : \n");

while(choice != 5){

printf("1. Push\n2. Pop\n3. Peek\n4. Display\n5. Exit");

scanf("%d", &choice);

switch (choice){

case 1:

printf("Enter element to push : ");

scanf("%d", &pushval);

push(pushval);

printf("Value pushed.");

break;

case 2:

popval = pop();

printf("%d\n", popval);

break;

case 3:

peek();

break;

case 4:

display();

break;

case 5:

break;

deafult:

printf("Invalid option!\n");

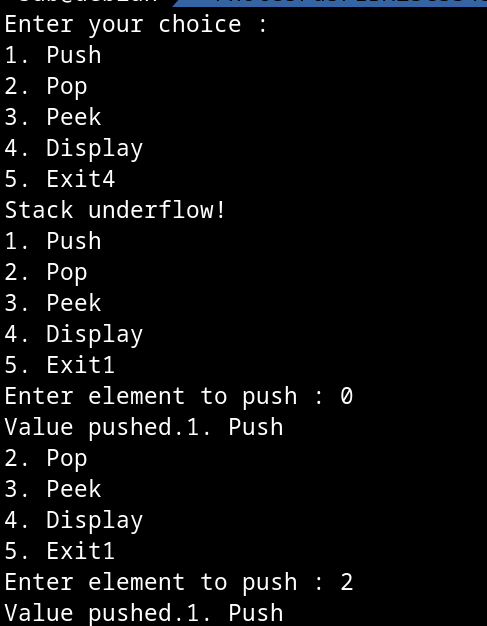
}

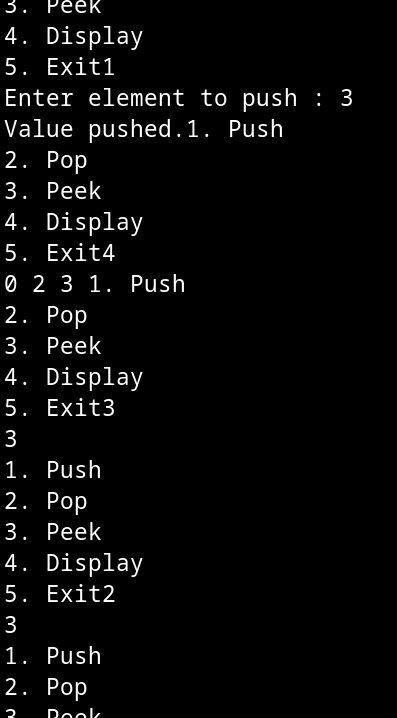
}

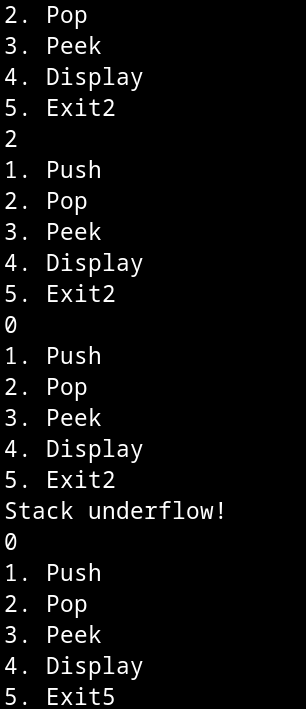
return 0;

}

Output :



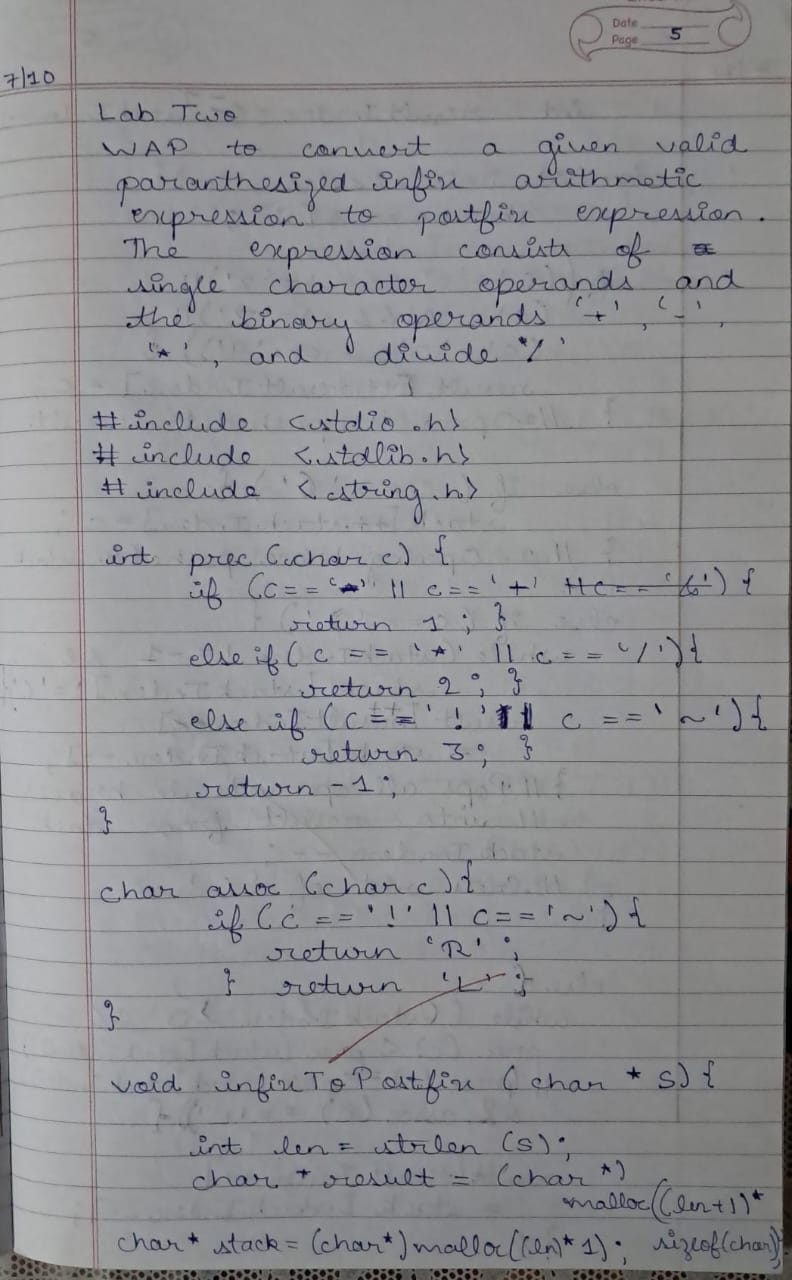


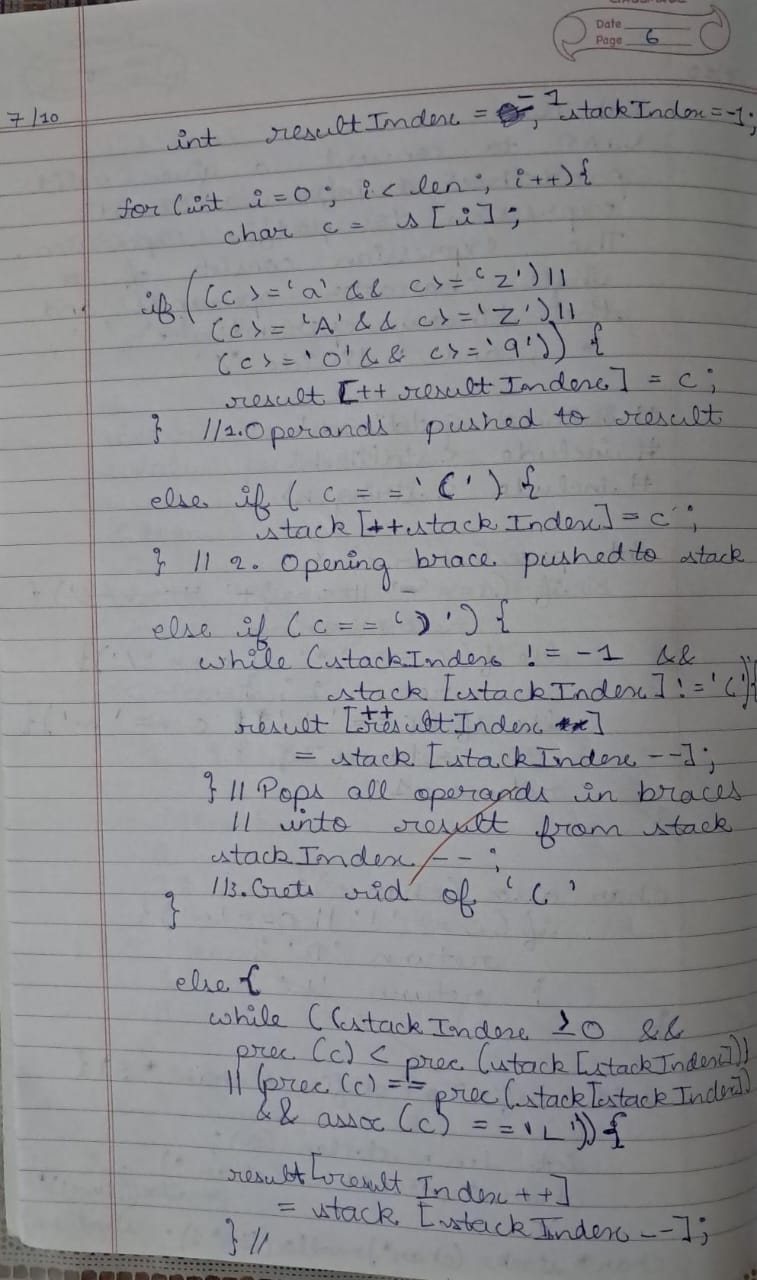


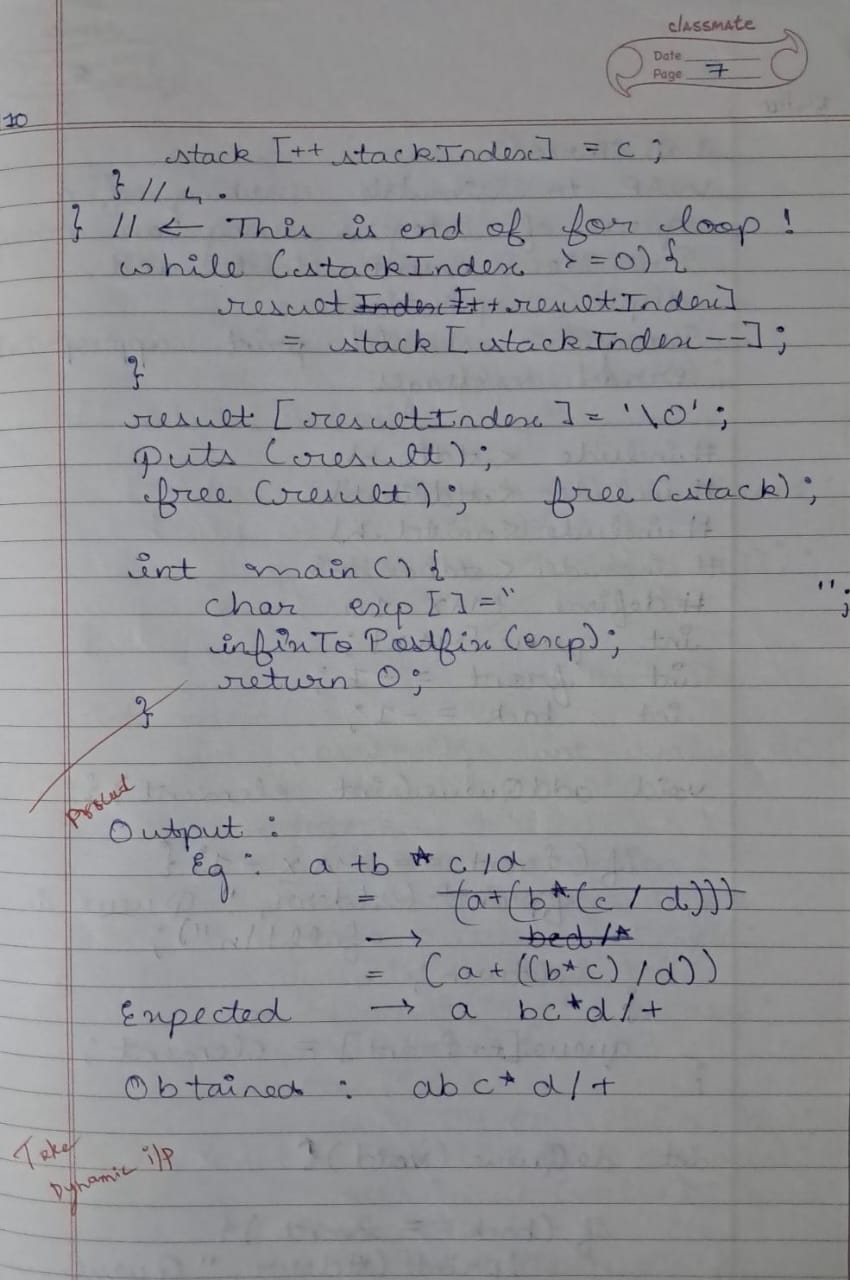
**Program 2**

WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), \* (multiply) and / (divide)

Observatiom :







Code :

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

#define MAX 10

int stack[MAX];

int top = -1;

bool isEmpty(){

if(top == -1){

return true;

} return false;

}

bool isFull(){

return top == (MAX-1);

}

void push(int value){

if(isFull()){

return;

}

stack[++top] = value;

}

int pop(){

if(isEmpty()){

return -1;

}

return stack[top--];

}

int getPrecedence(char ch){

switch (ch){

case '^':

return 2;

case '\*':

case '/':

case '%':

return 1;

case '+':

case '-':

return 0;

}

}

char getAssociativity(char ch){

if(ch == '^'){

return 'R';

} return 'L';

}

void \*infixToPostfic(char \* str){

char res[MAX], stk[MAX];

int rptr = -1;

int sptr = -1;

int len = strlen(str);

for(int i = 0; i < len; i++){

char ch = str[i];

if( // Case 1 : an operand

ch >= 'a' && ch <= 'z'

|| ch >= 'A' && ch <= 'Z'

|| ch >= '0' && ch <= '9'

){

res[++rptr] = ch;

}

else if( // Case 2 : (

ch == '('

){

stk[++sptr] = ch;

}

else if( // Case 3 : )

ch == ')'

){

while(sptr != -1 | stk[sptr] != '('){

res[++rptr] = str[sptr--];

} sptr--;

}

else{ // Case 4 : Operand

while(

sptr >= 0 && getPrecedence(ch) < getPrecedence(stk[sptr])

|| sptr >= 0 && getPrecedence(ch) == getPrecedence(stk[sptr]) && getAssociativity(ch) == 'L'

){

res[++rptr] = stk[sptr--];

} stk[++sptr] = ch;

}

}

while(sptr >= 0){

res[++rptr] = stk[sptr--];

}

res[++rptr] = '\0';

puts(res);

}

int main(){

char ch[MAX];

printf("Enter your expression :\n");

fgets(ch, MAX, stdin);

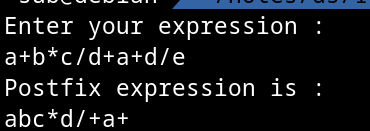
printf("Postfix expression is : \n");

infixToPostfic(ch);

return 0;

}

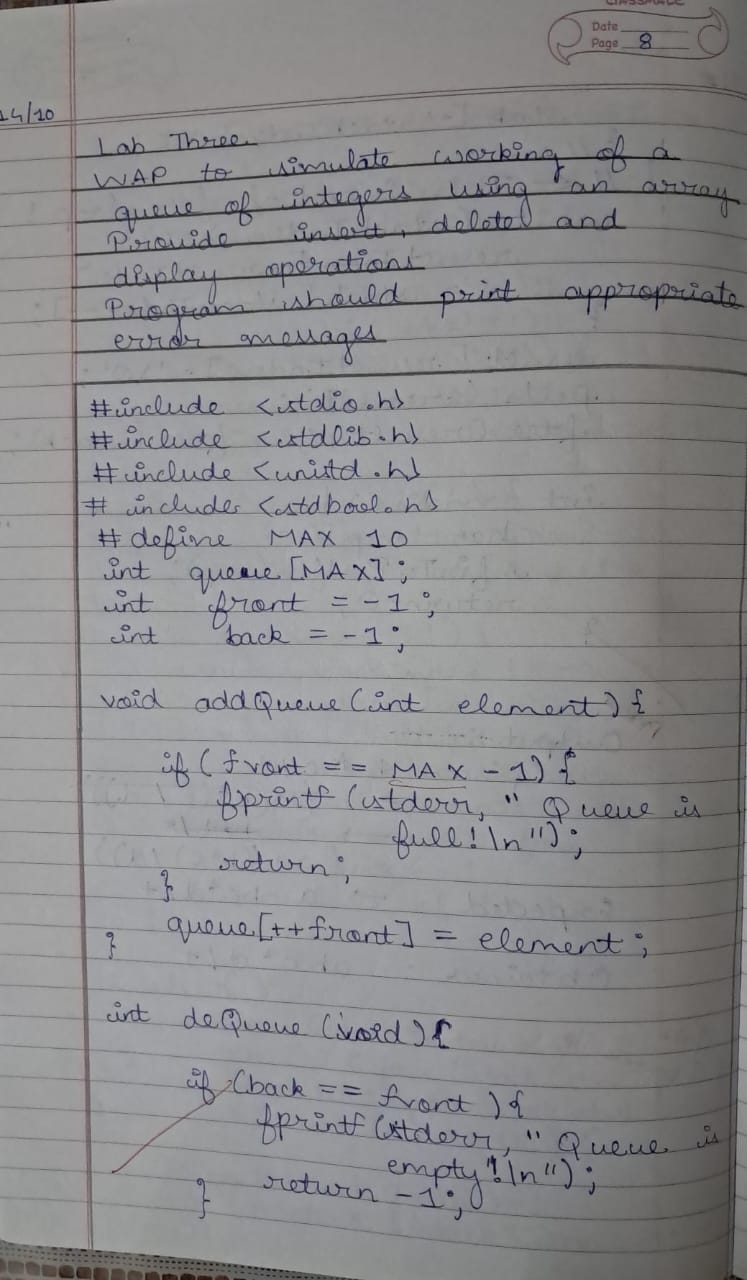
Output :

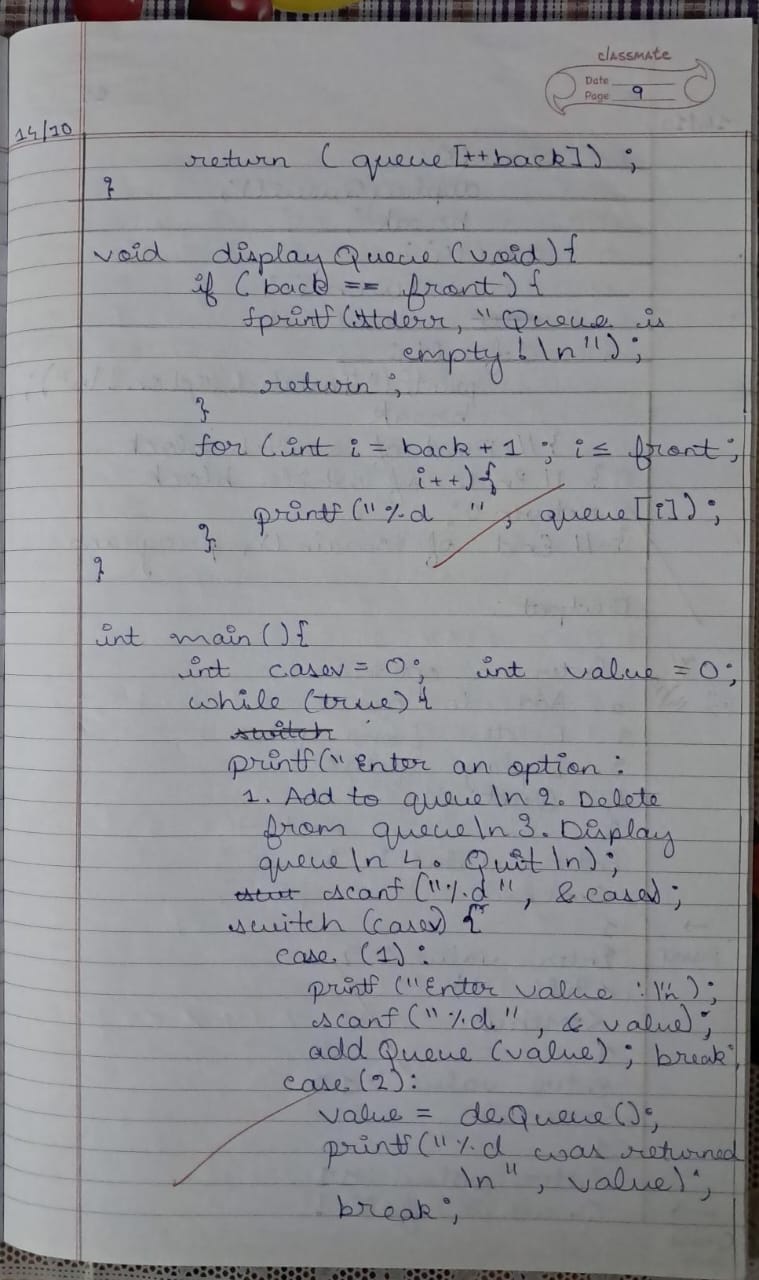


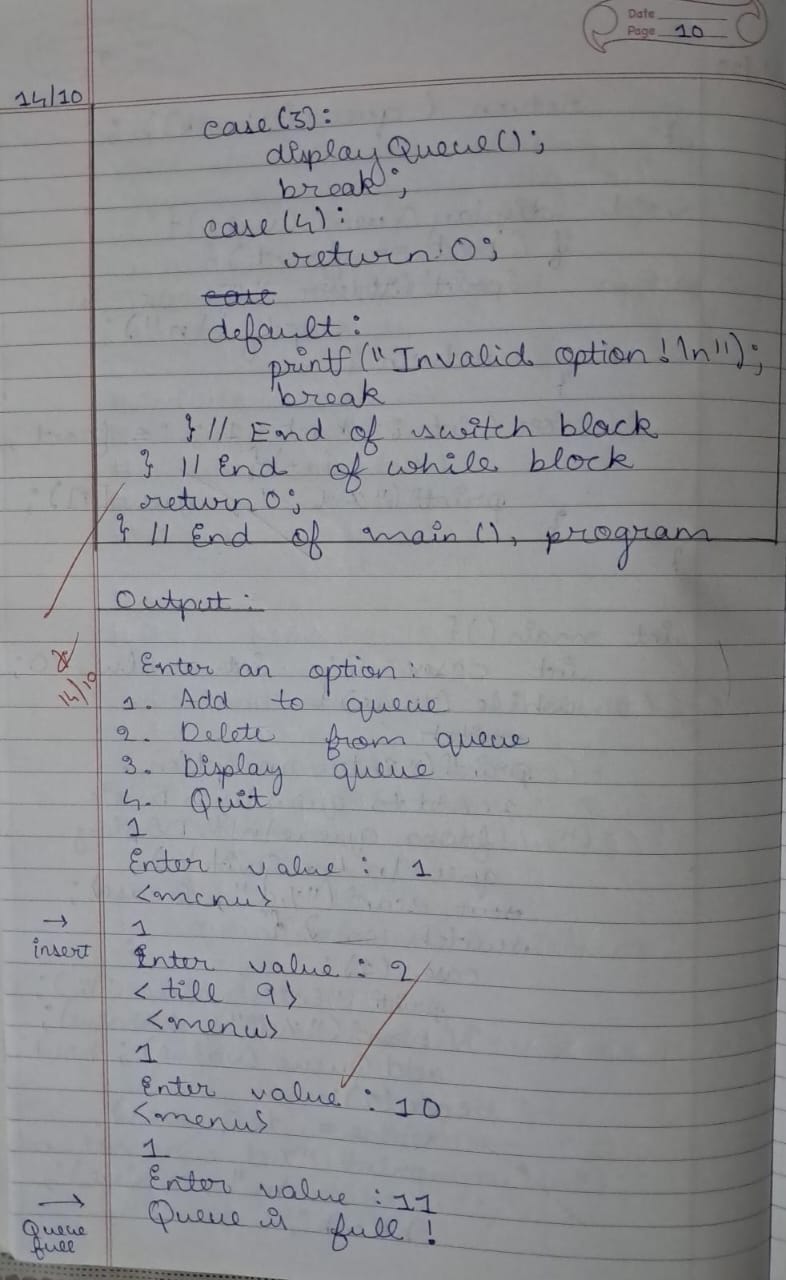
**Program 3**

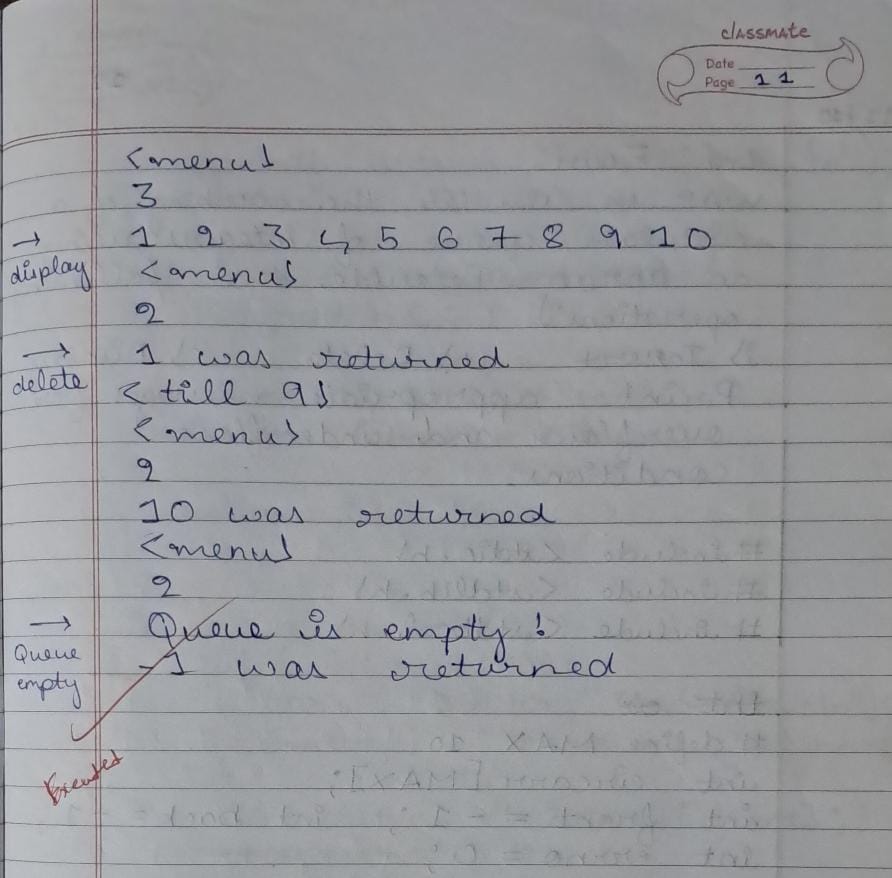
WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display The program should print appropriate messagesfor queue empty and queue overflow conditions

Observatiom :









Code :

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <stdbool.h>

#define MAX 10

int queue[MAX];

int front = -1;

int back = -1;

void addQueue(int element){

if(front == MAX - 1){

fprintf(stderr, "Queue is full!\n");

return;

}

queue[++front] = element;

}

int deQueue(void){

if(back == front){

fprintf(stderr, "Queue is empty!\n");

return -1;

}

return (queue[++back]);

}

void displayQueue(void){

if(back == front){

fprintf(stderr, "Queue is empty!\n");

return;

}

for(int i = back + 1; i <= front; i++){

printf("%d ", queue[i]);

}

}

int main(){

int casev = 0;

int value = 0;

while(true){

printf("Enter an option :\n1. Add to queue\n2. Delete from queue\n3. Display queue\n4. Quit\n");

scanf("%d", &casev);

switch (casev){

case 1:

printf("Enter value\n");

scanf("%d", &value);

addQueue(value);

break;

case 2:

value = deQueue();

printf("%d was returned\n", value);

break;

case 3:

displayQueue();

break;

case 4:

return 0;

case 5:

printf("Invalid option!\n");

break;

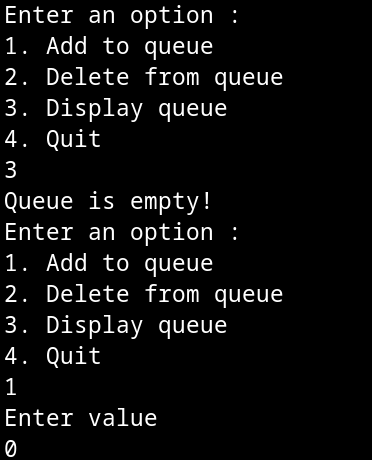
} // End of switch block

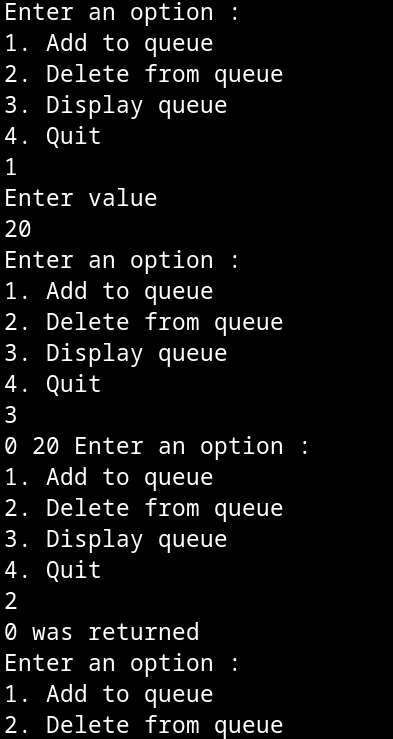
} // End of while block

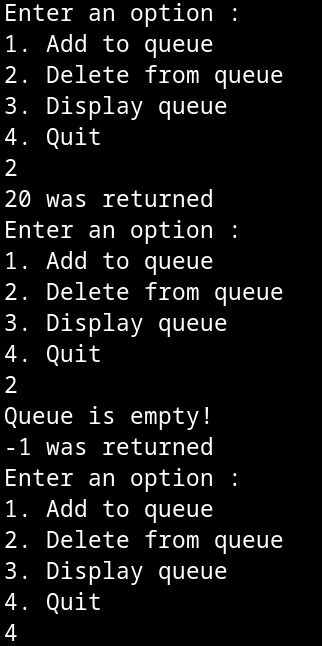
return 0;

} // End of main(), program

Output :



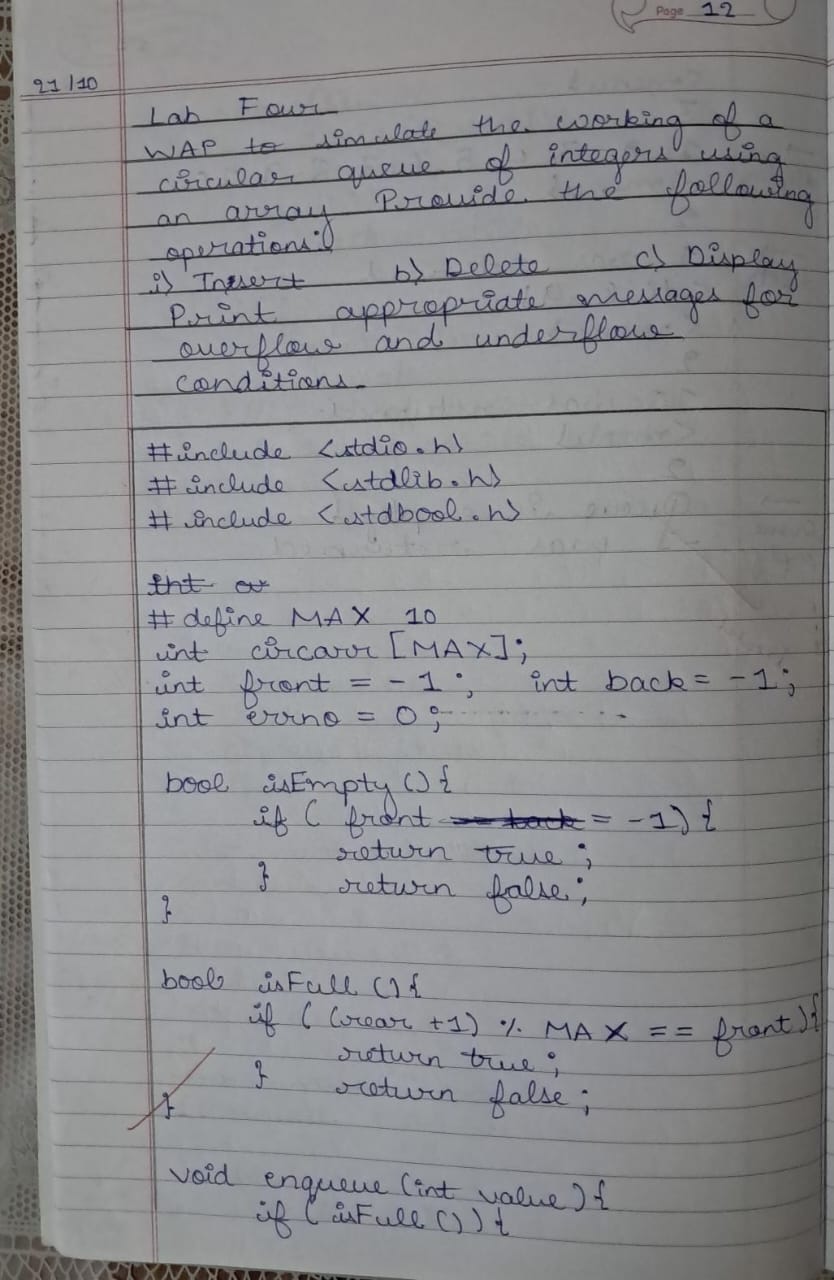


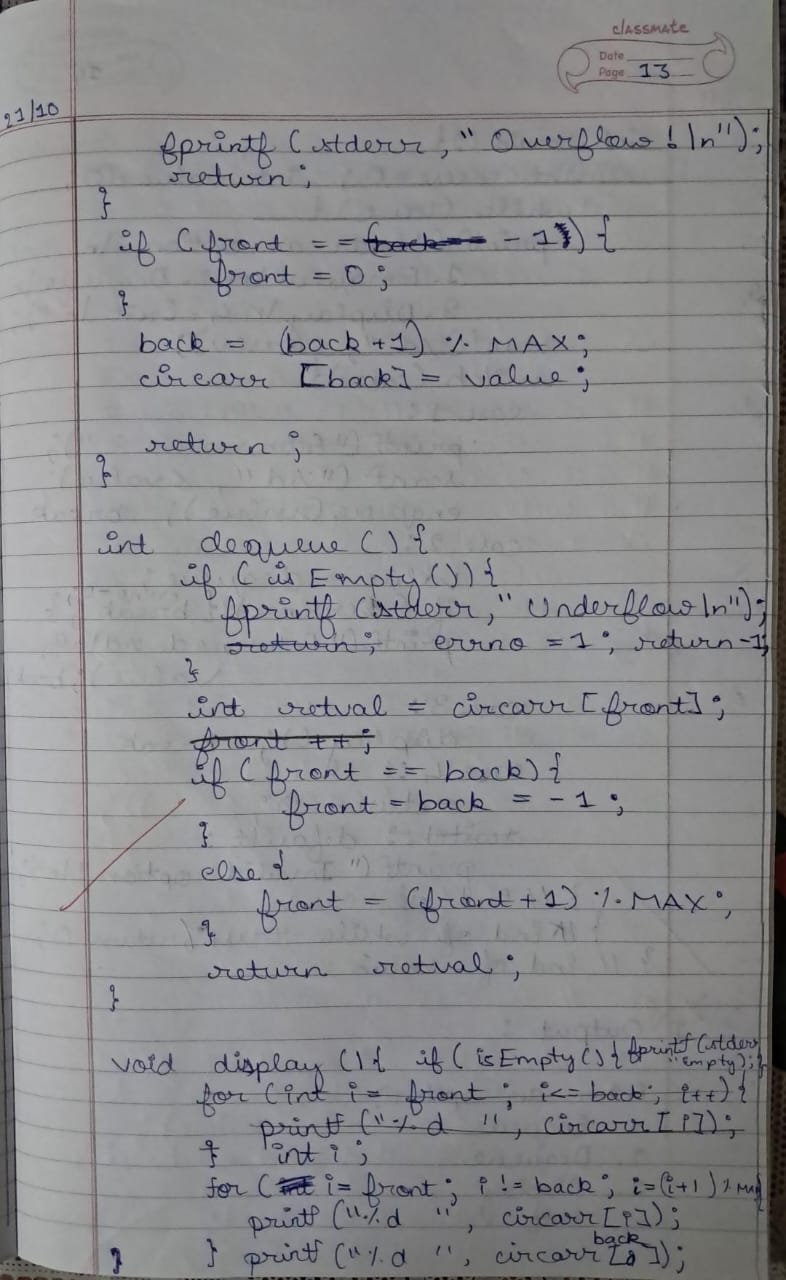


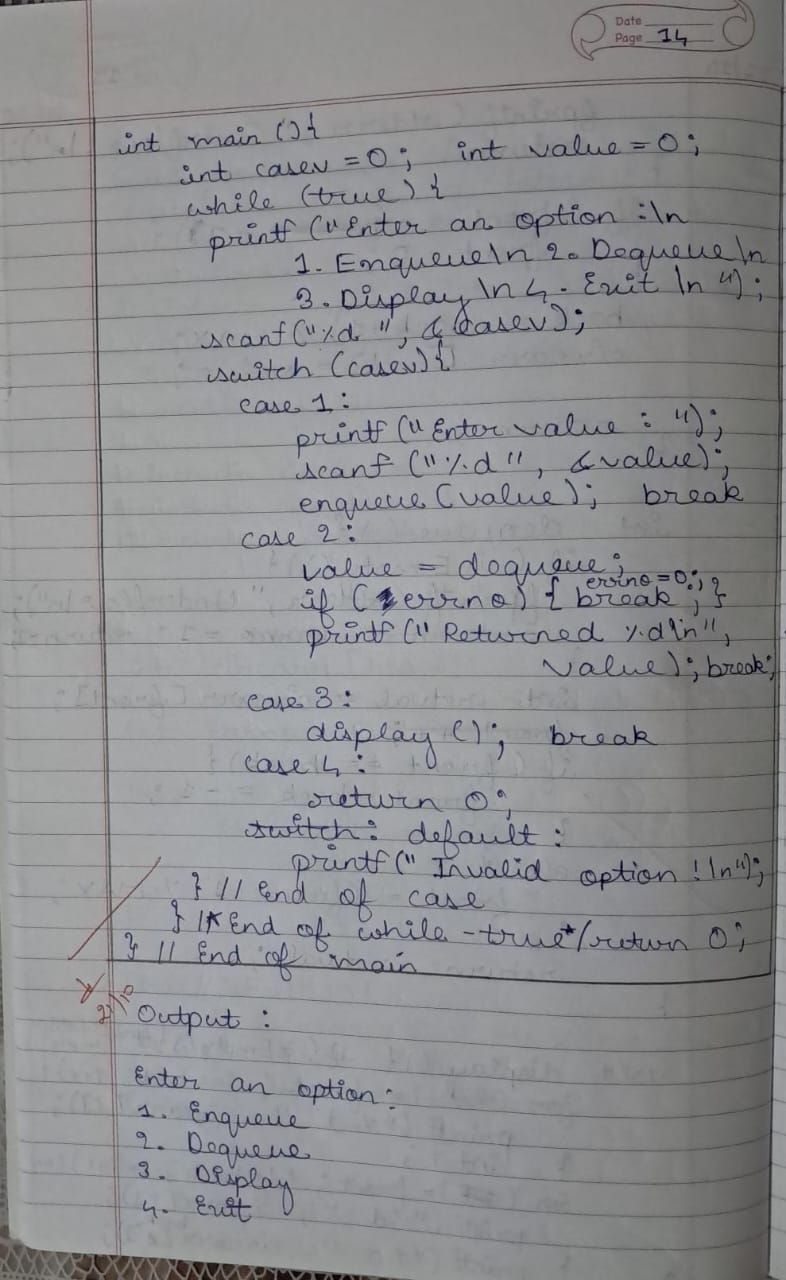
**Program 4**

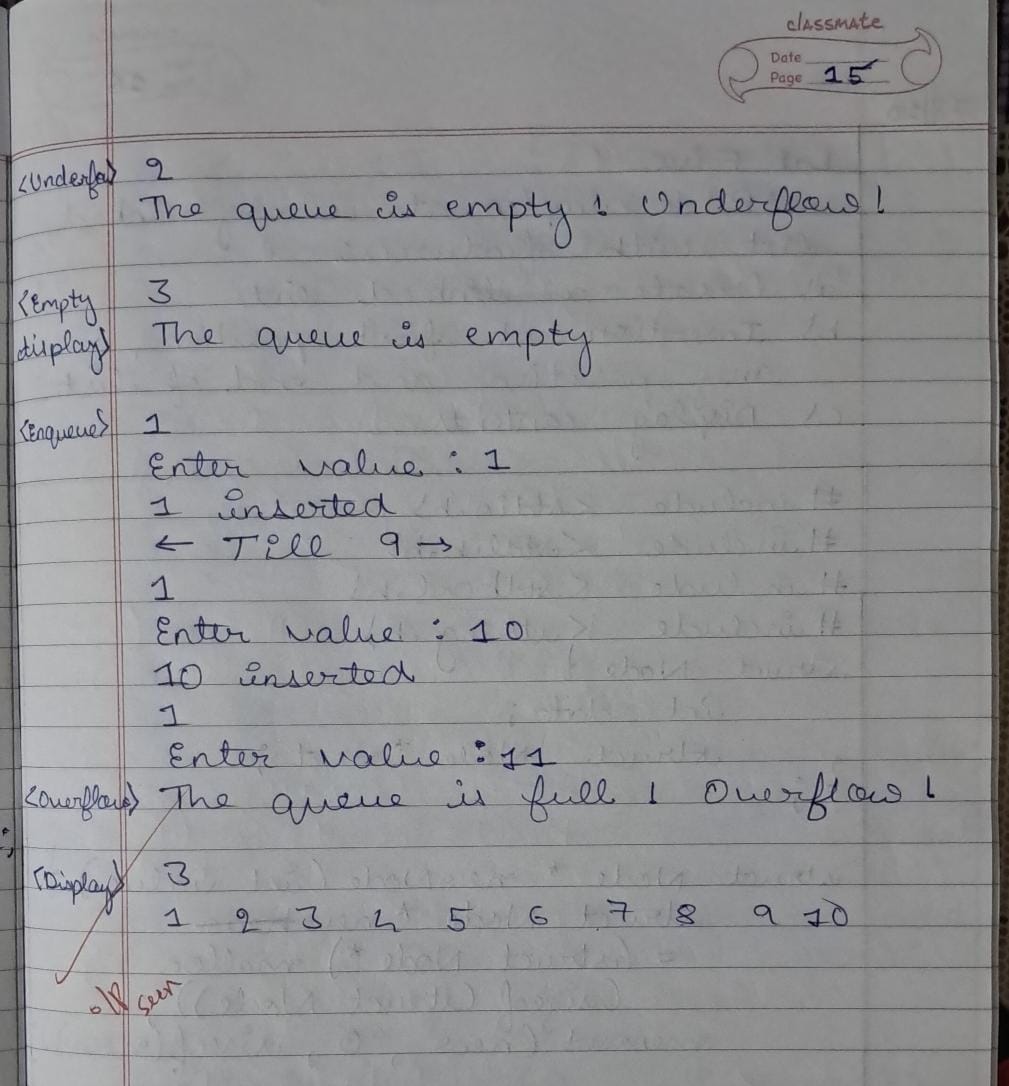
WAP to simulate the working of a queue of integers using an array. Provide the following operations: Insert, Delete, Display The program should print appropriate messagesfor queue empty and queue overflow conditions

Observatiom :









Code :

/\*\*

\* WAP to simulate the working of a circular queue of

\* integers using an array. Provide the following operations:

\* Insert, Delete & Display

\* The program should print appropriate messages for queue

\* empty and queue overflow conditions in C

\*/

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define MAX 10

#define DB (printf("control reached here\n"))

int circArr[MAX];

int front = -1;

int back = -1;

int myerrno = 0;

bool isFull(){

if( (back + 1) % MAX == front ){

return true;

}

return false;

}

bool isEmpty(){

if( front == -1){

return true;

}

return false;

}

void enqueue(int data){

if( isFull() ){

fprintf(stderr, "The queue is full! Overflow\n");

return;

}

if( front == -1 ){

front = 0;

}

back= (back+ 1) % MAX;

circArr[back] = data;

printf("%d inserted.\n", data);

return;

}

int dequeue(){

if( isEmpty() ){

fprintf(stderr, "The queue is empty! Underflow!\n");

myerrno = 1;

return -1;

}

int retval = circArr[front];

if( front == back){

front = back = -1;

}

else{

front = (front + 1) % MAX;

}

return retval;

}

void display(){

if( isEmpty() ){

fprintf(stderr, "The queue is empty!\n");

return;

}

int i;

for(int i = front; i != back; i = (i+1) % MAX){

printf("%d ", circArr[i]);

} printf("%d \n", circArr[back]);

}

int main(){

int casev = 0;

int value = 0;

while(true){

printf("\nEnter an option :\n1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n");

scanf("%d", &casev);

switch(casev){

case 1:

printf("Enter value : ");

scanf("%d", &value);

enqueue(value);

break;

case 2:

value = dequeue();

if(myerrno){

myerrno = 0;

break;

}

printf("Returned value is %d\n", value);

break;

case 3:

display();

break;

case 4:

return 0;

default:

printf("Invalid option!\n");

break;

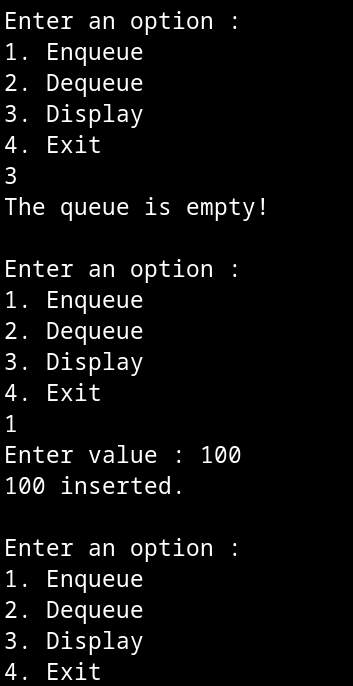
} // End of switch case

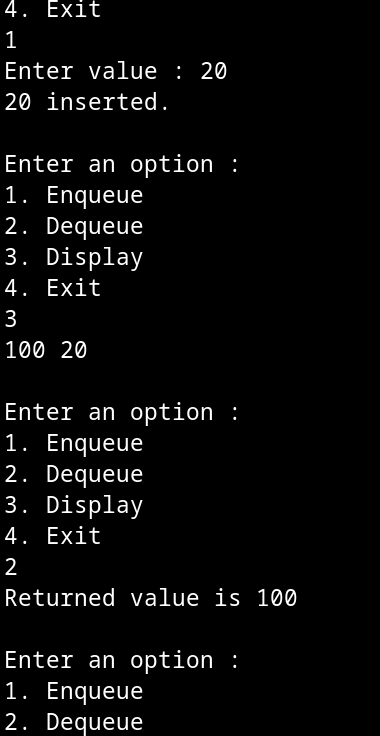
} // End of while-true

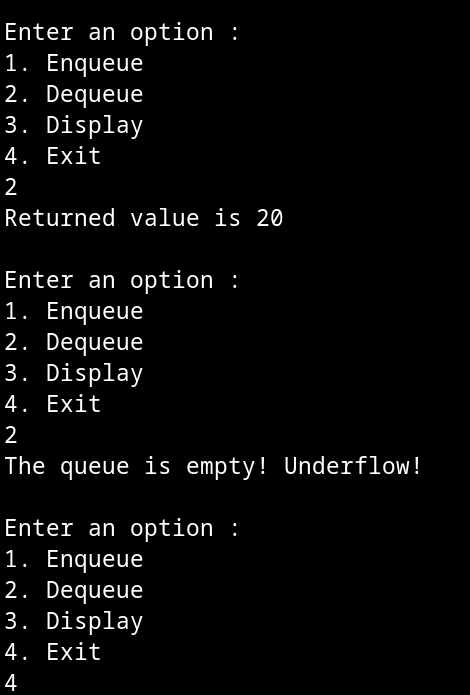
return 0;

} // End of main

Output :



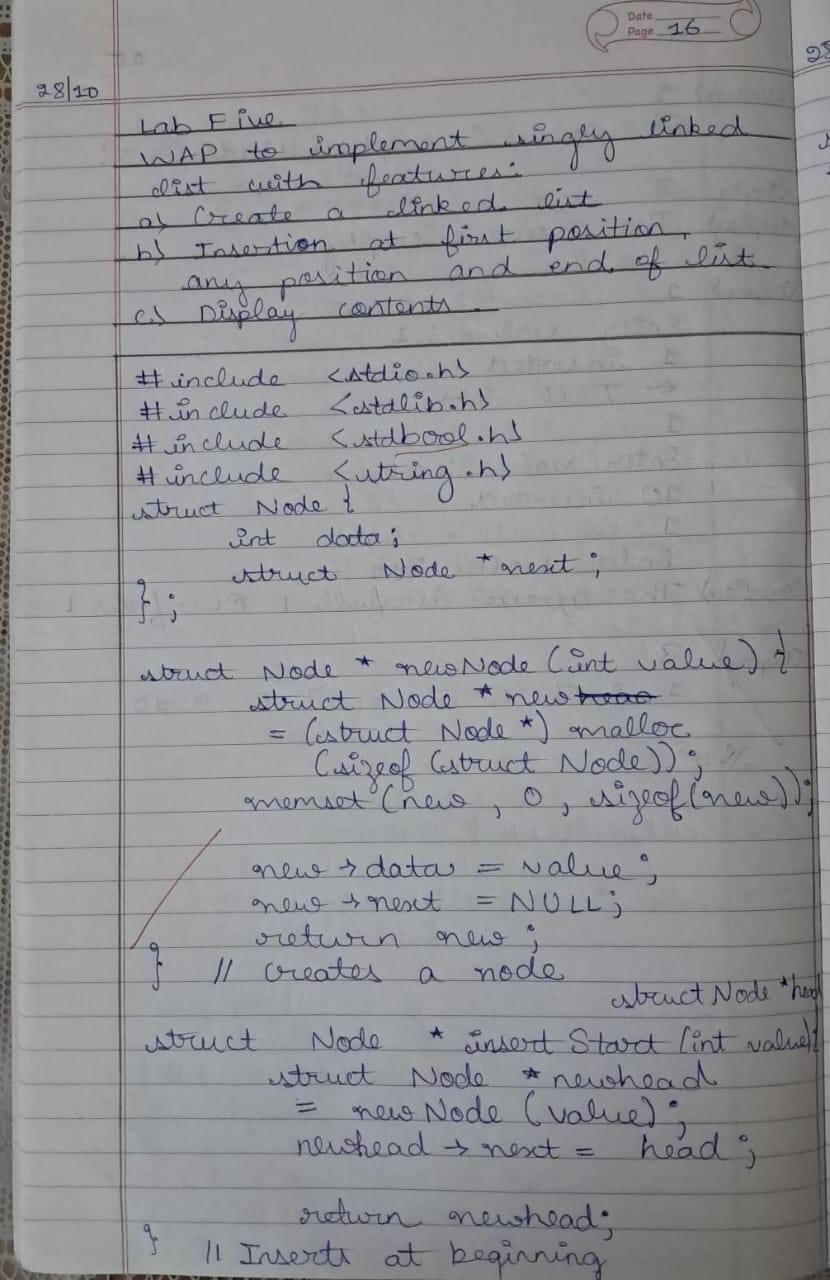


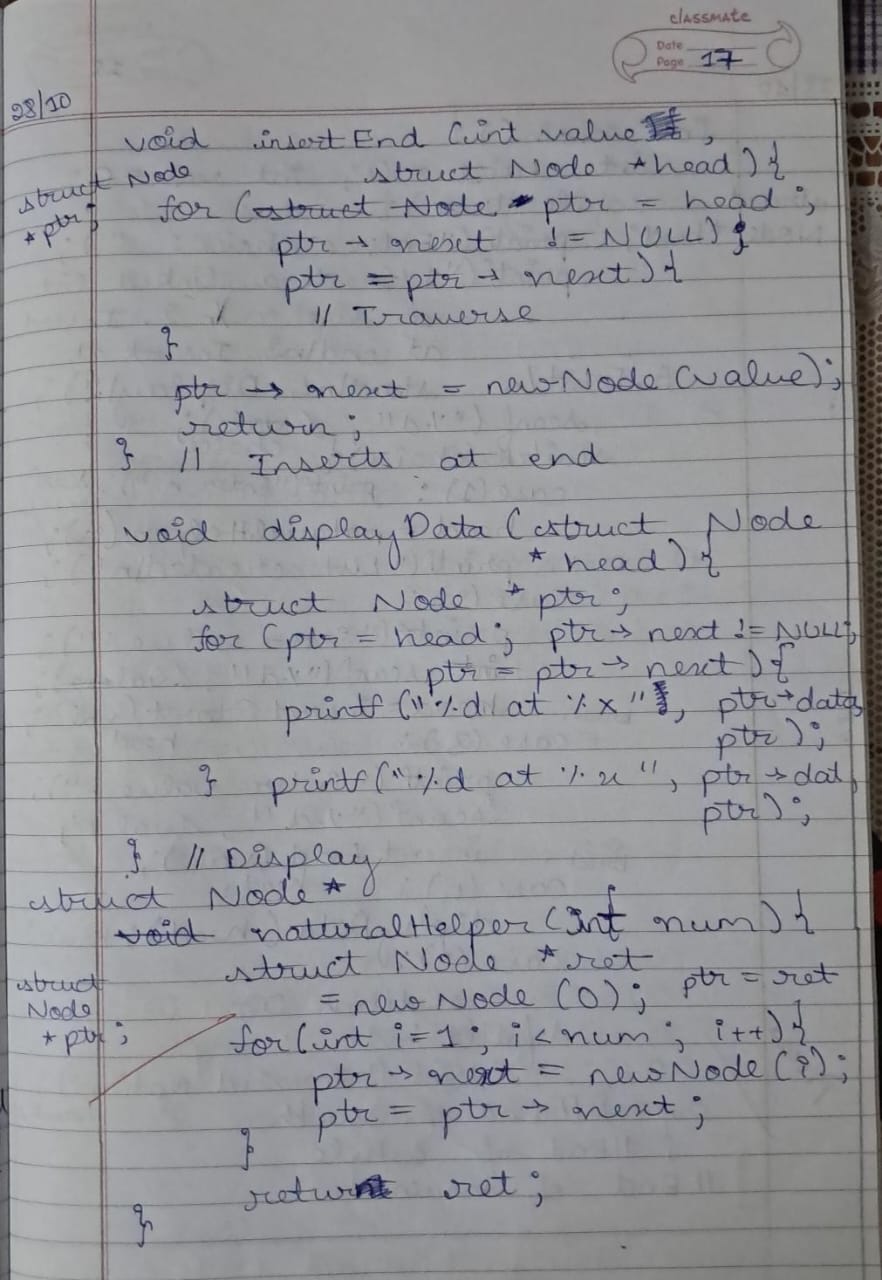


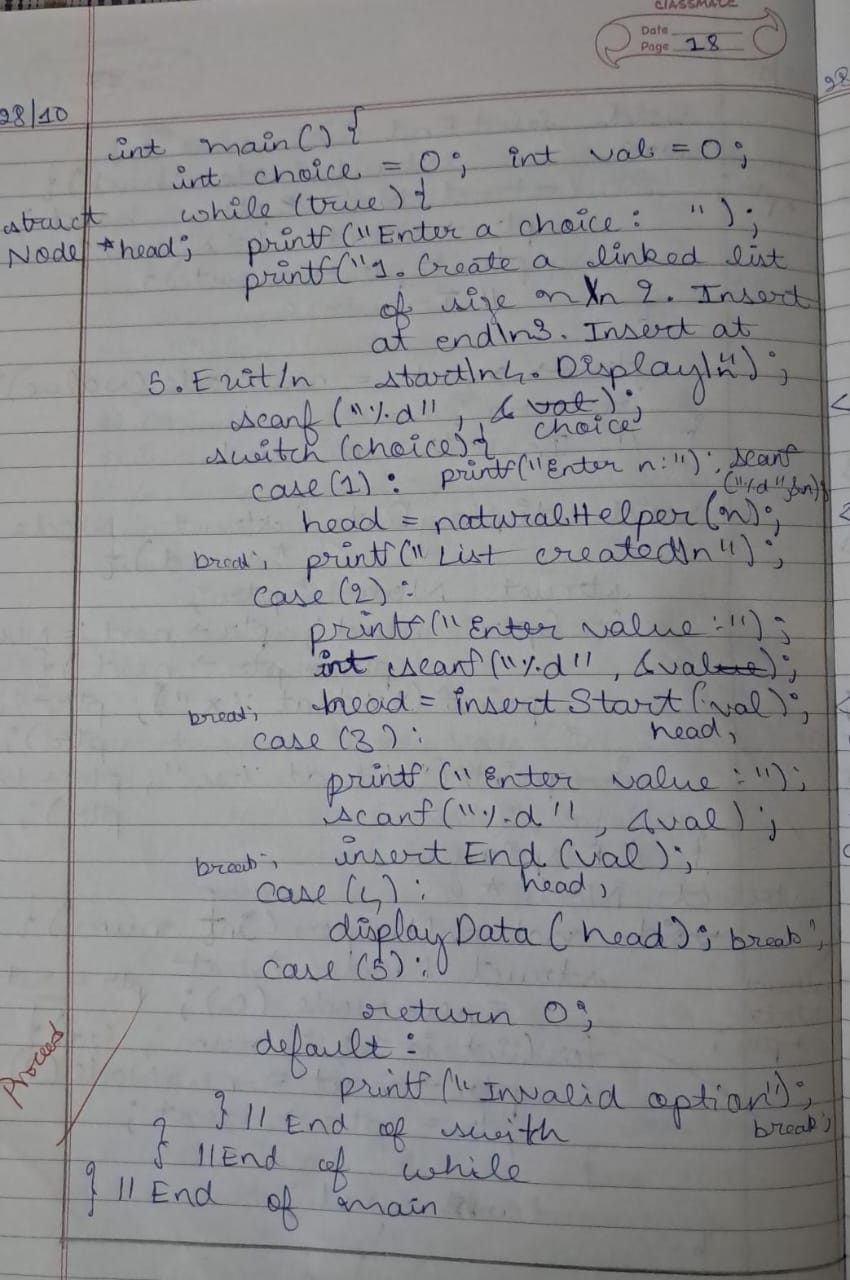
**Program 5**

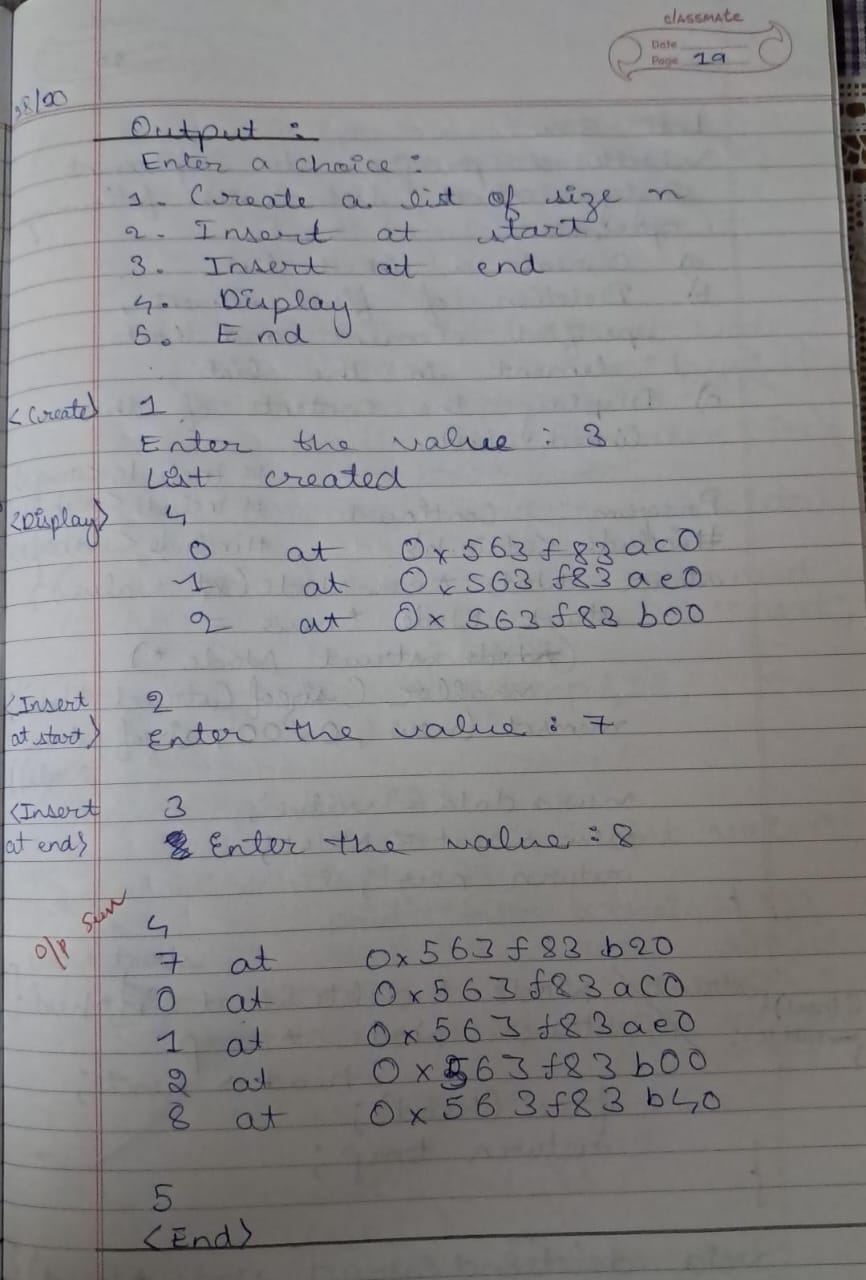
WAP to Implement Singly Linked List with following operations a) Createalinkedlist. b) Insertion of a node at first position, at any position and at end of list. Display the contents of the linked list.

Observatiom :









Code :

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

struct Node{

int data;

struct Node \*next;

};

struct Node \*newNode(int value){

struct Node \*new = (struct Node \*) malloc(sizeof(struct Node));

memset(new, 0, sizeof(struct Node));

new->data = value;

new->next = NULL;

return new;

} // Create a new node

struct Node \*insertStart(struct Node \*head, int value){

struct Node \*newhead = newNode(value);

newhead->next = head;

return newhead;

} // Inserts at begenning

void insertEnd(struct Node \*head, int value){

struct Node \*ptr;

for(ptr = head; ptr->next != NULL; ptr = ptr->next){

// Traverse

}

ptr->next = newNode(value);

return;

} // Inserts at beginning

void displayData(struct Node \*head){

if(!head){

printf("Empty linked list!\n");

return;

}

struct Node \*ptr;

for(ptr = head; ptr->next != NULL; ptr = ptr->next){

printf("%d at %p\n", ptr->data, ptr);

} printf("%d at %p\n\n", ptr->data, ptr);

} // Display

struct Node \*naturalHelper(int num){

struct Node \*head;

struct Node \*ptr = newNode(0);

head = ptr;

for(int i = 1; i < num; i++){

ptr->next = newNode(i);

ptr = ptr->next;

}

return head;

}

int main(){

int choice = 0;

int val = 0;

struct Node \*head;

while(true){

printf("Enter a choice : \n");

printf("1. Create a linked list of size n\n2. Insert at end\n3. Insert at start\n4. Display\n5. Exit\n");

scanf("%d", &choice);

switch (choice){

case (1):

printf("Enter n : ");

scanf("%d", &val);

head = naturalHelper(val);

break;

case (2):

printf("Enter value : ");

scanf("%d", &val);

head = insertStart(head, val);

break;

case (3):

printf("Enter value : ");

scanf("%d", &val);

insertEnd(head, val);

break;

case (4):

displayData(head);

break;

case (5):

return 0;

default:

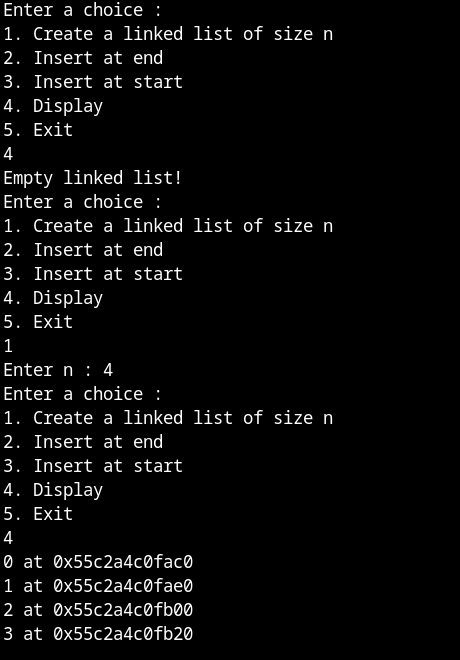
printf("Invalid option\n");

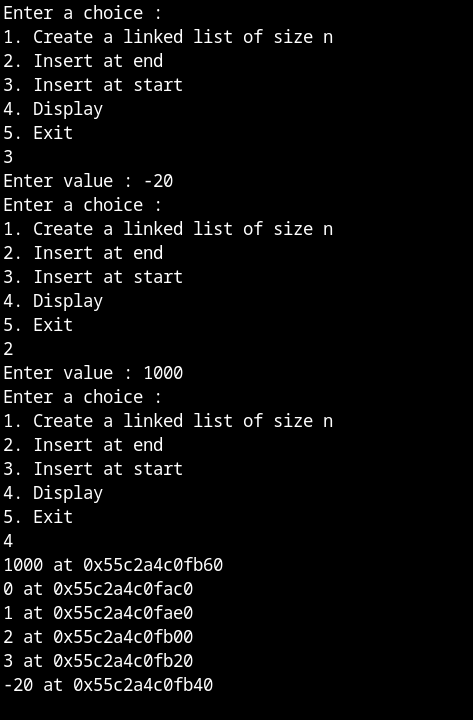
} // End of switch

} // End of while-loop

} // End of main

Output :





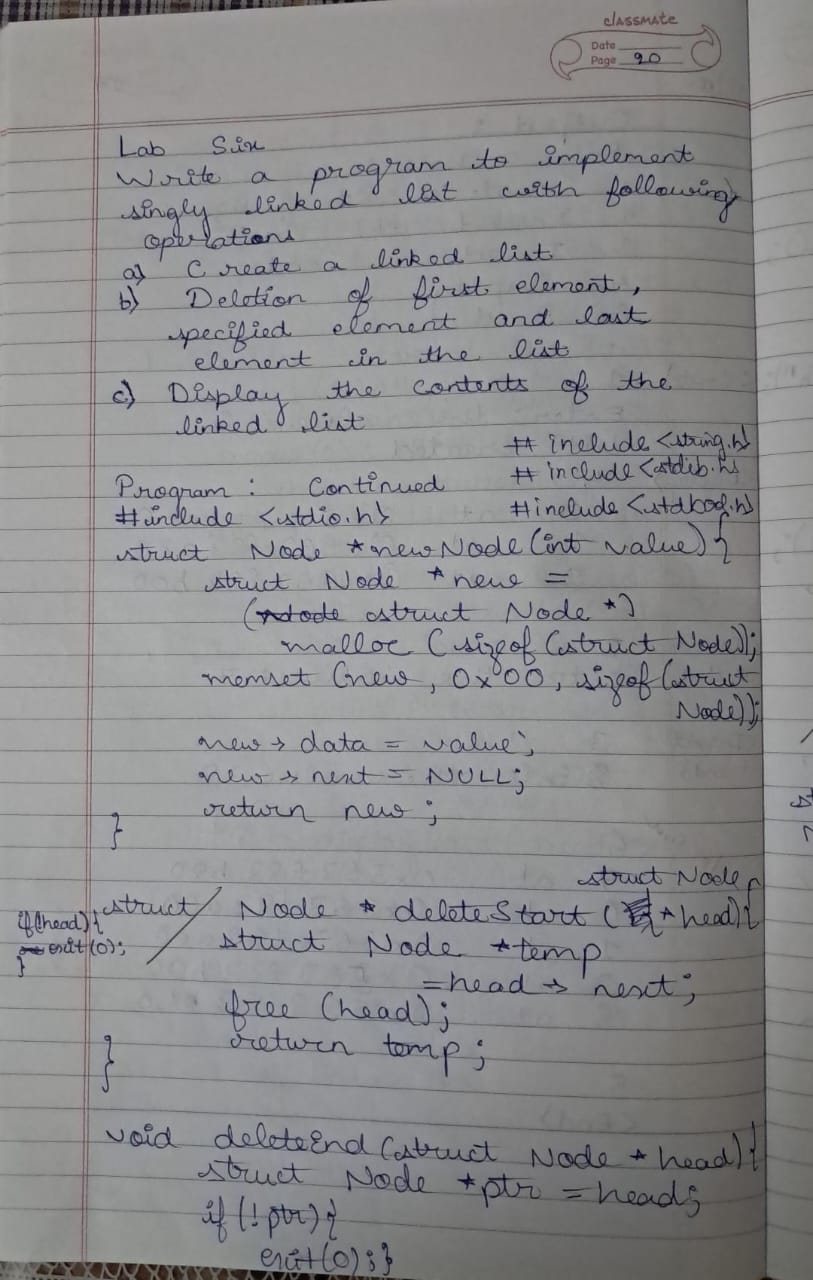
Leetcode problem : Valid Parantheses

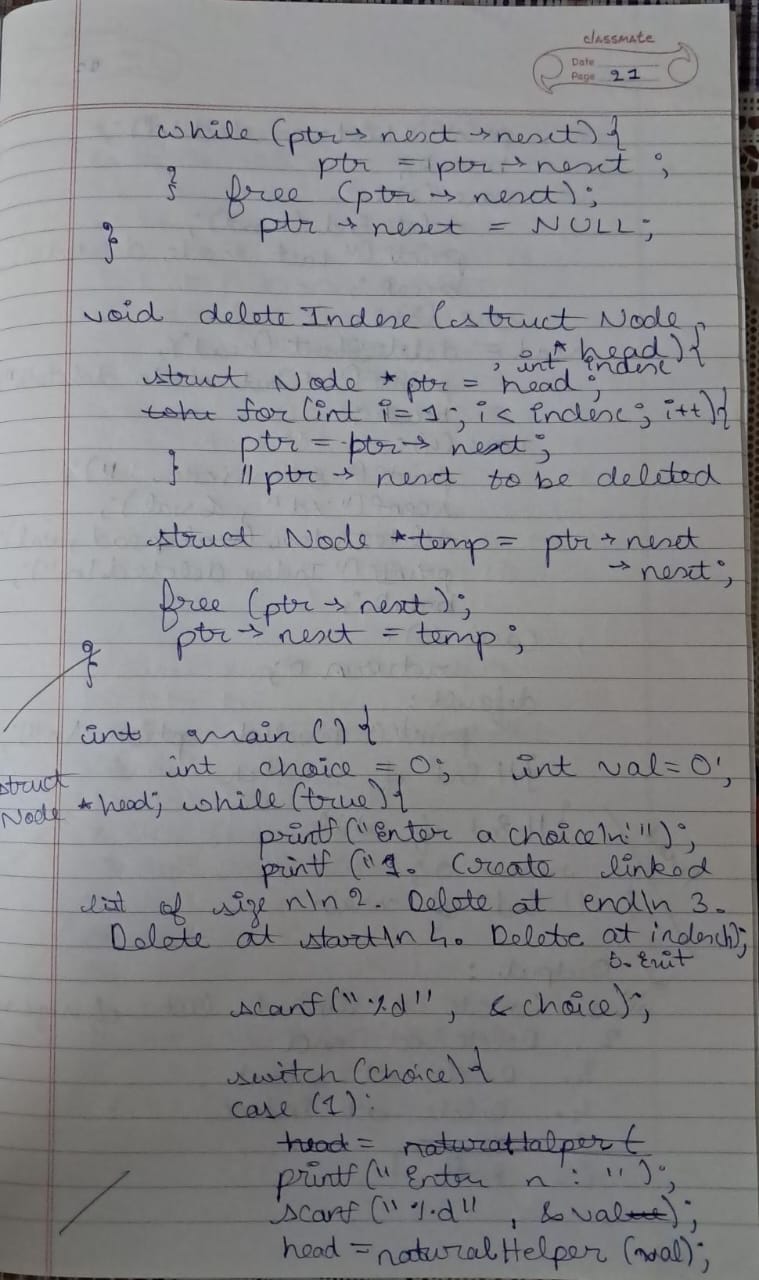


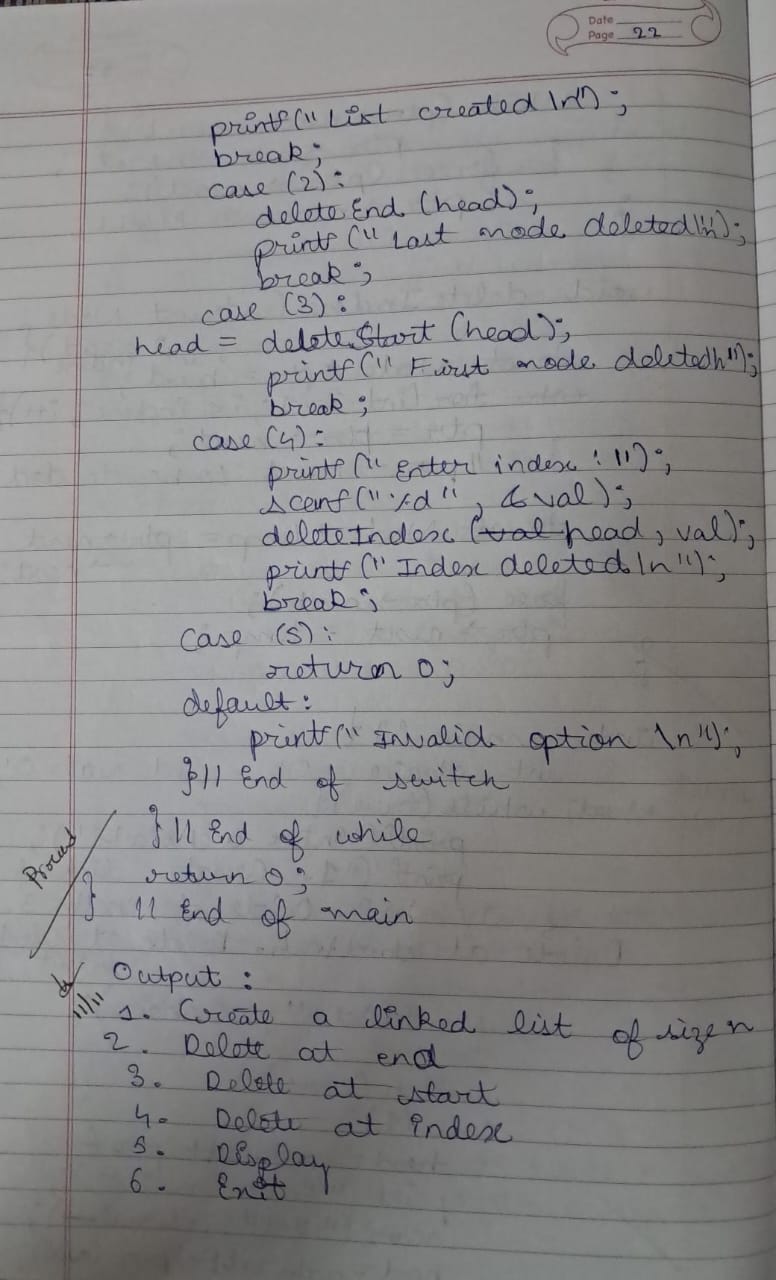
**Program 6**

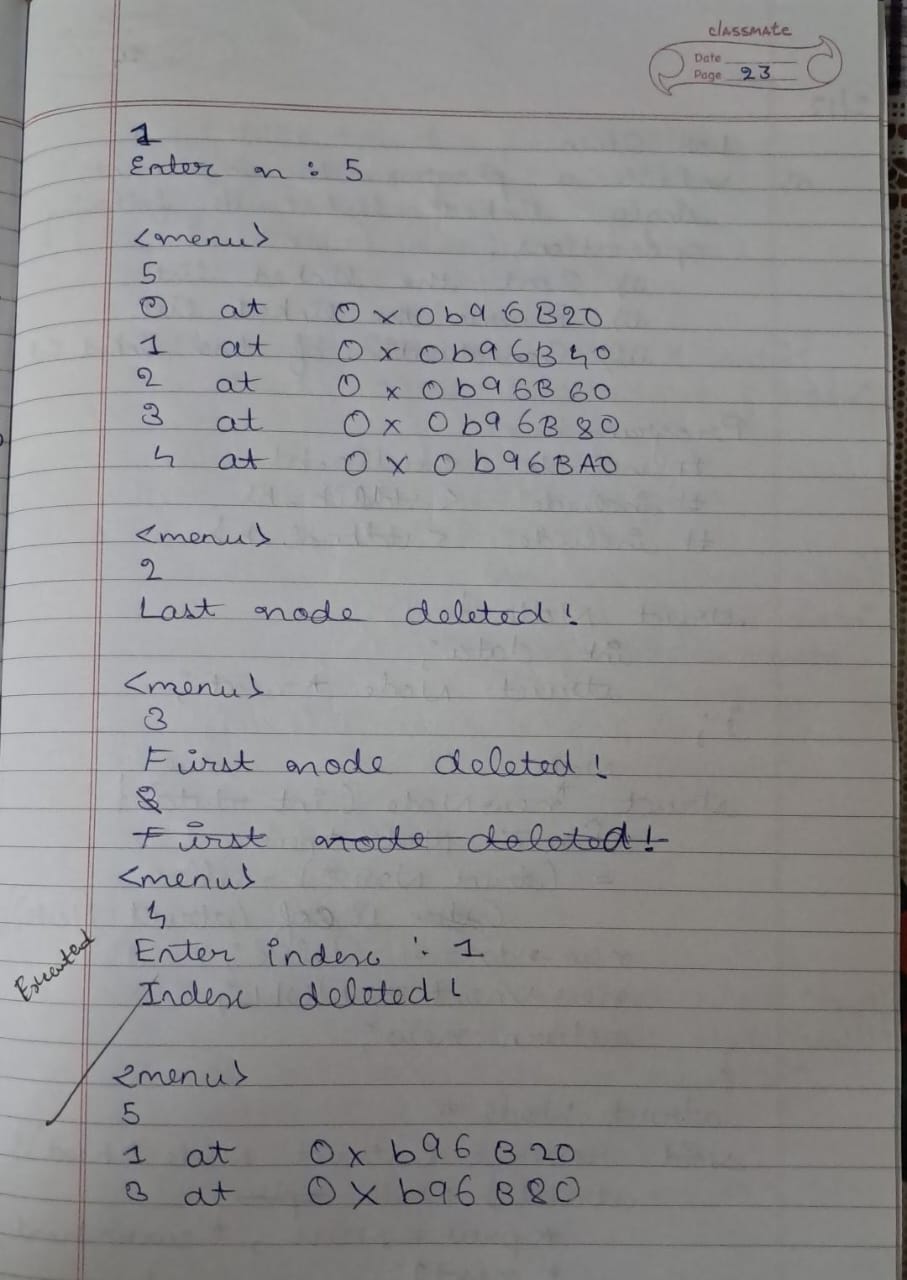
WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

Observatiom :









Code :

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <string.h>

struct Node{

int data;

struct Node \*next;

};

struct Node \*newNode(int value){

struct Node \*new = (struct Node \*) malloc(sizeof(struct Node));

memset(new, 0, sizeof(struct Node));

new->data = value;

new->next = NULL;

return new;

} // Create a new node

struct Node \*insertStart(struct Node \*head, int value){

struct Node \*newhead = newNode(value);

newhead->next = head;

return newhead;

} // Inserts at begenning

void insertEnd(struct Node \*head, int value){

struct Node \*ptr;

for(ptr = head; ptr->next != NULL; ptr = ptr->next){

// Traverse

}

ptr->next = newNode(value);

return;

} // Inserts at beginning

void displayData(struct Node \*head){

if(!head){

printf("Linked list empty!\n");

return;

}

struct Node \*ptr;

for(ptr = head; ptr->next != NULL; ptr = ptr->next){

printf("%d at %p\n", ptr->data, ptr);

} printf("%d at %p\n", ptr->data, ptr);

} // Display

struct Node \*naturalHelper(int num){

struct Node \*head = newNode(0);

struct Node \*ptr = head;

for(int i = 1; i < num; i++){

ptr->next = newNode(i);

ptr = ptr->next;

}

return head;

}

struct Node \*deleteStart(struct Node \*head){

struct Node \*ptr = head->next;

free(head);

return ptr;

}

void \*deleteEnd(struct Node \*head){

struct Node \*ptr = head;

while(ptr->next->next){

ptr = ptr->next;

}

free(ptr->next);

ptr->next = NULL;

}

void deleteIndex(struct Node \*head, int index){

struct Node \*ptr = head;

for(int i = 1; i < index; i++){

ptr = ptr->next;

}

struct Node \*temp = ptr->next->next;

free(ptr->next);

ptr->next = temp;

return;

}

int main(){

int choice = 0;

int val = 0;

struct Node \*head;

while(true){

printf("Enter your choice :\n1. Create a linked list of size n\n2. deleteStart\n3. deleteEnd\n4. display\n5. Exit\n");

scanf("%d", &choice);

switch (choice){

case (1):

printf("Enter n : ");

scanf("%d", &val);

head = naturalHelper(val);

break;

case (2):

head = deleteStart(head);

break;

case (3):

deleteEnd(head);

break;

case (4):

displayData(head);

break;

case (5):

return 0;

default:

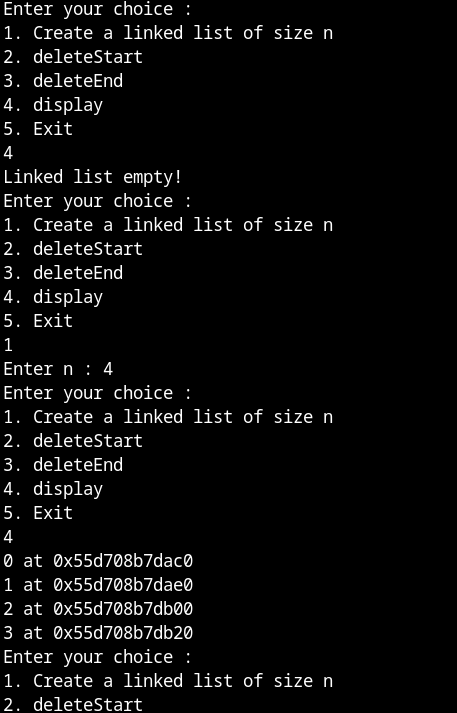
printf("Invalid option\n");

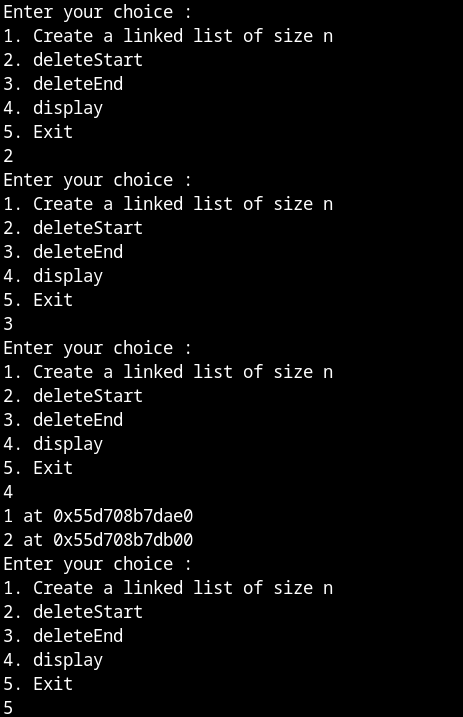
} // End of switch

} // End of while-loop

} // End of main

Output :





Leetcode problem : Daily Temperatures