**MANAV RACHNA INTERNATIONAL INSTITUTE OF RESEARCH AND STUDIES DATA STRUCTURE USING C LAB FILE**

**4.5CA151C01**

**NAME – MANYATA GOSWAMI**

**ROLL NUMBER – 24/SCA/BCA(AI&ML)/037**

Q1. Write a program in C to implement insertion in 1D Array . Ans. Input :–

#include <stdio.h>

void insertElement(int arr[], int \*size, int element, int position) { if (position < 0 || position > \*size) {

printf("Invalid position!\n"); return;

}

for (int i = \*size; i > position; i--) { arr[i] = arr[i - 1];

}

arr[position] = element; (\*size)++;

}

int main() {

int arr[100], size, element, position; printf("Enter the size of the array: "); scanf("%d", &size);

printf("Enter %d elements of the array: ", size); for (int i = 0; i < size; i++) {

scanf("%d", &arr[i]);

}

printf("Enter the element to insert: "); scanf("%d", &element);

printf("Enter the position (0-based index): "); scanf("%d", &position);

insertElement(arr, &size, element, position); printf("Array after insertion: ");

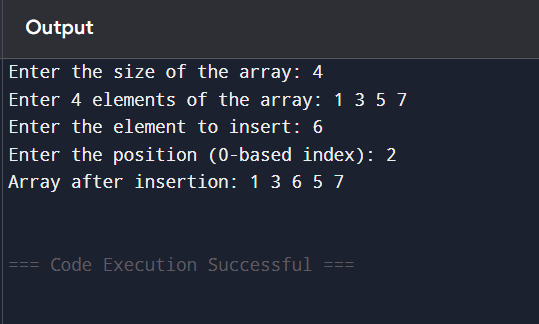
for (int i = 0; i < size; i++) { printf("%d ", arr[i]);

}

printf("\n"); return 0;

}

Output :-



Q2. Write a program in C to implement deletion in 1D Array . Ans. Input :-

#include <stdio.h>

void deleteElement(int arr[], int \*size, int pos) { if (pos < 0 || pos >= \*size) {

printf("Invalid position! Please enter a valid index (0 to %d).\n", \*size - 1); return;

}

for (int i = pos; i < \*size - 1; i++) { arr[i] = arr[i + 1];

}

(\*size)--;

}

int main() {

int arr[100], n, pos;

printf("Enter the number of elements in the array: "); scanf("%d", &n);

printf("Enter %d elements:\n", n); for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

printf("Enter the position (0-based index) of the element to delete: "); scanf("%d", &pos);

deleteElement(arr, &n, pos);

printf("Array after deletion:\n"); for (int i = 0; i < n; i++) {

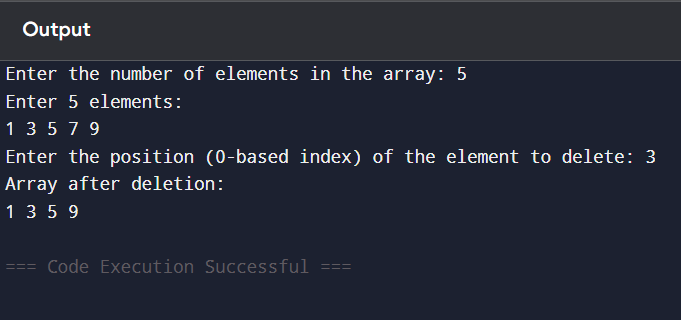
printf("%d ", arr[i]);

}

return 0;

}

Output :-



Q3. Write a program in C to concatenate two array . Ans. Input :-

#include <stdio.h>

void concatenateArrays(int arr1[], int size1, int arr2[], int size2, int result[]) { int i, j;

for (i = 0; i < size1; i++) { result[i] = arr1[i];

}

for (j = 0; j < size2; j++) { result[i + j] = arr2[j];

}

}

void displayArray(int arr[], int size) { for (int i = 0; i < size; i++) {

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

}

printf("\n");

}

int main() {

int size1, size2;

printf("Enter size of first array: "); scanf("%d", &size1);

int arr1[size1];

printf("Enter elements of first array: ");

for (int i = 0; i < size1; i++) { scanf("%d", &arr1[i]);

}

printf("Enter size of second array: "); scanf("%d", &size2);

int arr2[size2];

printf("Enter elements of second array: "); for (int i = 0; i < size2; i++) {

scanf("%d", &arr2[i]);

}

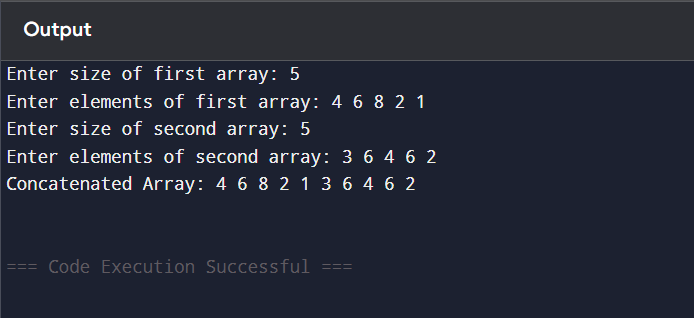
int result[size1 + size2];

concatenateArrays(arr1, size1, arr2, size2, result); printf("Concatenated Array: ");

displayArray(result, size1 + size2); return 0;

}

Output :-



Q4. Write a program in c to implement the following operations on 2d array ( addition, subtraction, multiplication, transpose ) .

Ans. Input :- #include <stdio.h> #define ROW 3

#define COL 3

void inputMatrix(int matrix[ROW][COL], char name) {

printf("Enter elements of matrix %c (%dx%d):\n", name, ROW, COL); for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) {

printf("%c[%d][%d]: ", name, i, j);

scanf("%d", &matrix[i][j]);

}

}

}

void displayMatrix(int matrix[ROW][COL]) { for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) { printf("%d\t", matrix[i][j]);

}

printf("\n");

}

}

void addMatrices(int A[ROW][COL], int B[ROW][COL], int result[ROW][COL]) { for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) { result[i][j] = A[i][j] + B[i][j];

}

}

}

void subtractMatrices(int A[ROW][COL], int B[ROW][COL], int result[ROW][COL]) {

for (int i = 0; i < ROW; i++) { for (int j = 0; j < COL; j++) {

result[i][j] = A[i][j] - B[i][j];

}

}

}

void multiplyMatrices(int A[ROW][COL], int B[ROW][COL], int result[ROW][COL]) {

for (int i = 0; i < ROW; i++) { for (int j = 0; j < COL; j++) {

result[i][j] = 0;

for (int k = 0; k < COL; k++) { result[i][j] += A[i][k] \* B[k][j];

}

}

}

}

void transposeMatrix(int A[ROW][COL], int result[ROW][COL]) { for (int i = 0; i < ROW; i++) {

for (int j = 0; j < COL; j++) { result[j][i] = A[i][j];

}

}

}

int main() {

int A[ROW][COL], B[ROW][COL], result[ROW][COL];

inputMatrix(A, 'A');

inputMatrix(B, 'B');

printf("Matrix A:\n"); displayMatrix(A); printf("Matrix B:\n"); displayMatrix(B);

addMatrices(A, B, result); printf("Addition of Matrices:\n"); displayMatrix(result);

subtractMatrices(A, B, result); printf("Subtraction of Matrices:\n"); displayMatrix(result);

multiplyMatrices(A, B, result); printf("Multiplication of Matrices:\n"); displayMatrix(result);

transposeMatrix(A, result);

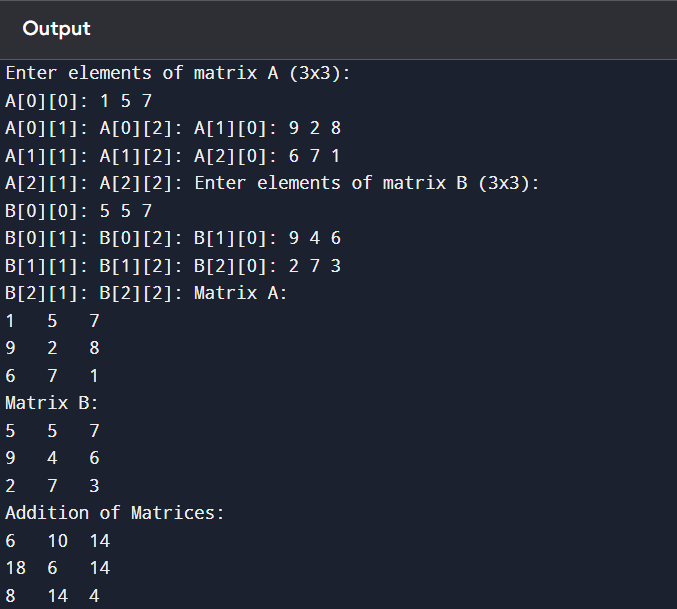
printf("Transpose of Matrix A:\n"); displayMatrix(result);

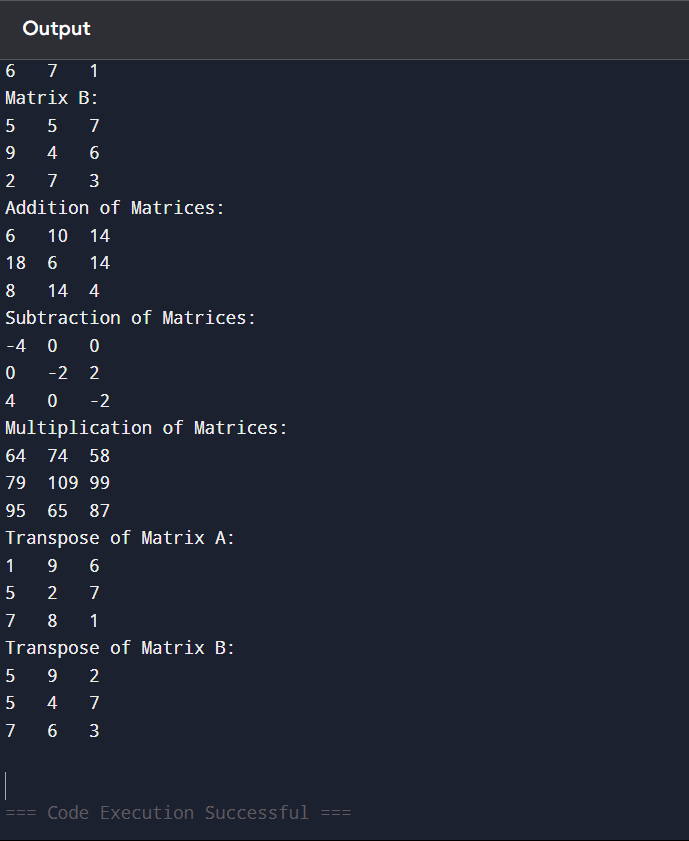
transposeMatrix(B, result); printf("Transpose of Matrix B:\n"); displayMatrix(result);

return 0;

}

Output :-





Q5. Write a program in C to implement operations on stack using array. Ans. Input :-

#include <stdio.h> #define MAX 10

int stack[MAX], top = -1; void push() {

int value;

if (top == MAX - 1) { printf("Stack Overflow!\n"); return;

}

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

stack[++top] = value;

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

}

void pop() {

if (top == -1) {

printf("Stack Underflow!\n"); return;

}

printf("%d popped from stack.\n", stack[top--]);

}

void display() { if (top == -1) {

printf("Stack is empty!\n");

return;

}

printf("Stack elements: "); for (int i = top; i >= 0; i--) {

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

}

printf("\n");

}

int main() { int choice; do {

printf("\nStack Operations:\n");

printf("1. Push\n2. Pop\n3. Display\n4. Exit\n"); printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: push(); break; case 2: pop(); break;

case 3: display(); break;

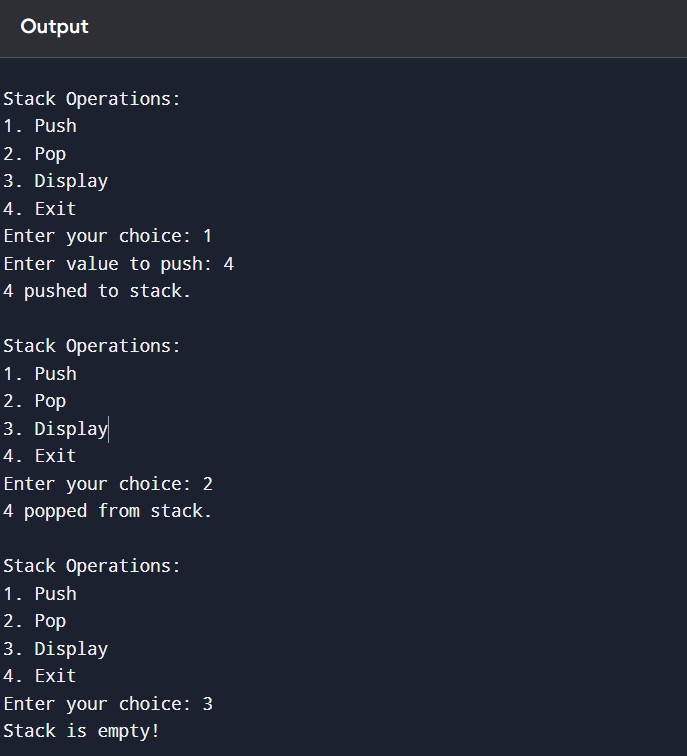
case 4: printf("Exiting...\n"); break; default: printf("Invalid choice!\n");

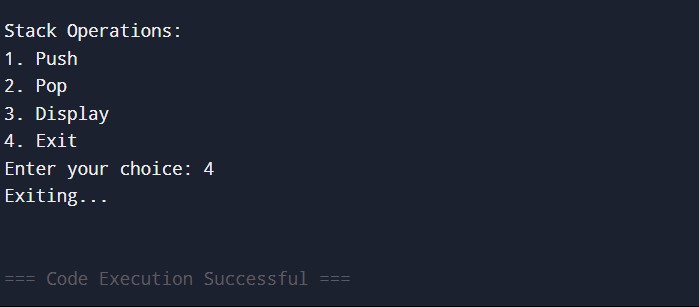
}

} while (choice != 4); return 0;

}

Output :-





Q6. Write a program in C to implement operations on queue using array. Ans. Input :-

#include <stdio.h> #define MAX 10

int queue[MAX], front = -1, rear = -1; void enqueue() {

int value;

if (rear == MAX - 1) { printf("Queue Overflow!\n"); return;

}

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

if (front == -1) front = 0; queue[++rear] = value;

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

}

void dequeue() {

if (front == -1 || front > rear) { printf("Queue Underflow!\n"); front = rear = -1;

return;

}

printf("%d dequeued from queue.\n", queue[front++]);

}

void display() {

if (front == -1 || front > rear) { printf("Queue is empty!\n"); return;

}

printf("Queue elements: ");

for (int i = front; i <= rear; i++) { printf("%d ", queue[i]);

}

printf("\n");

}

int main() { int choice; do {

printf("\nQueue Operations:\n");

printf("1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n"); printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: enqueue(); break; case 2: dequeue(); break; case 3: display(); break;

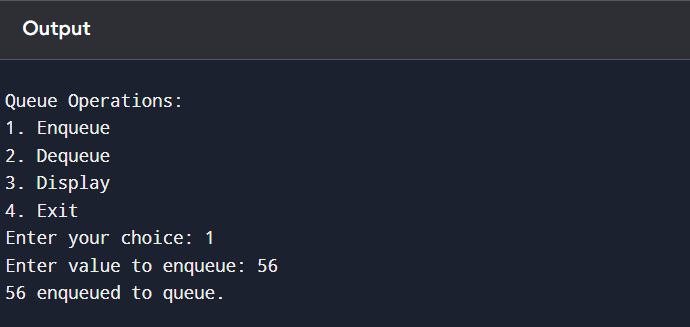
case 4: printf("Exiting...\n"); break; default: printf("Invalid choice!\n");

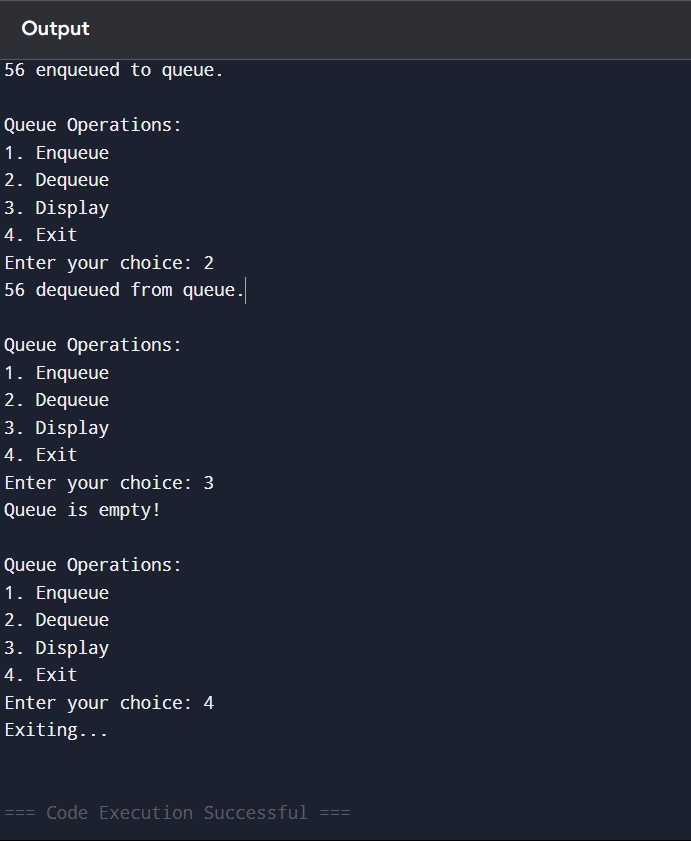
}

} while (choice != 4); return 0;

}

Output :-





Q7. Write a program in C to implement operations on circular queue using array.

Ans. Input :- #include <stdio.h> #define MAX 5

int queue[MAX], front = -1, rear = -1; void enqueue() {

int value;

if ((rear + 1) % MAX == front) { printf("Queue Overflow!\n"); return;

}

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

if (front == -1) front = 0; rear = (rear + 1) % MAX; queue[rear] = value;

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

}

void dequeue() { if (front == -1) {

printf("Queue Underflow!\n"); return;

}

printf("%d dequeued from queue.\n", queue[front]); if (front == rear) {

front = rear = -1;

} else {

front = (front + 1) % MAX;

}

}

void display() {

if (front == -1) {

printf("Queue is empty!\n"); return;

}

printf("Queue elements: "); int i = front;

while (1) {

printf("%d ", queue[i]); if (i == rear) break;

i = (i + 1) % MAX;

}

printf("\n");

}

int main() { int choice; do {

printf("\nCircular Queue Operations:\n");

printf("1. Enqueue\n2. Dequeue\n3. Display\n4. Exit\n"); printf("Enter your choice: ");

scanf("%d", &choice); switch (choice) {

case 1: enqueue(); break; case 2: dequeue(); break; case 3: display(); break;

case 4: printf("Exiting...\n"); break; default: printf("Invalid choice!\n");

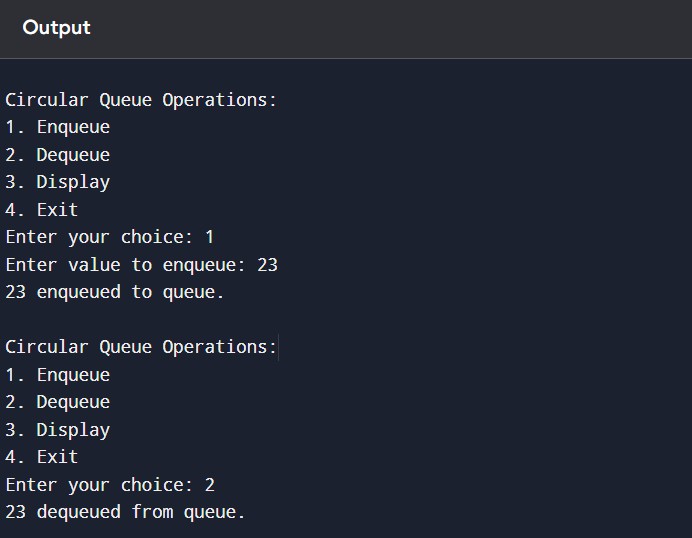
}

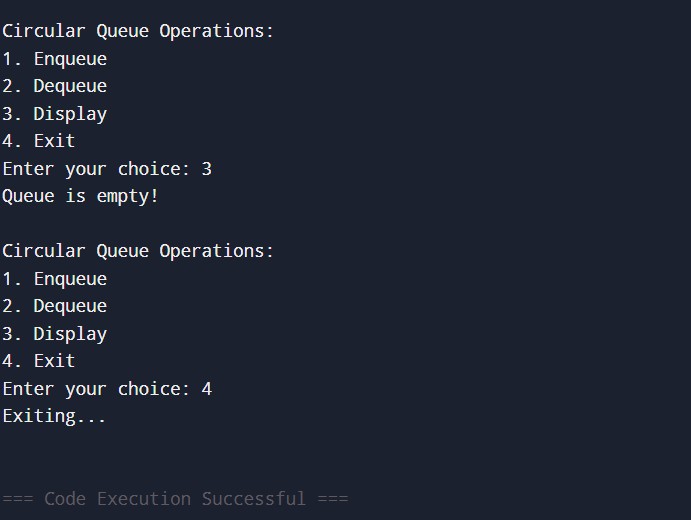
} while (choice != 4);

return 0;

}

Output :-





Q8. Perform insertion operation in link list( beginning, mid, end ) and perform deletion operation in link list ( beginning, mid , end) .

Ans. Input :- #include <stdio.h> #include <stdlib.h> typedef struct node {

int info;

struct node \*next;

} Node;

Node \*start = NULL; void insbeg();

void insmid(); void insend(); void delbeg(); void delmid(); void delend(); void display();

int main() { int ch, ch1; while (1)

{

printf("1. Insertion 2. Deletion 3. Display 4. Exit\n"); printf("Enter your choice: ");

scanf("%d", &ch); switch (ch) {

case 1:

printf("1. Begin 2. Middle 3. End\n"); printf("Enter your insertion choice: "); scanf("%d", &ch1);

switch (ch1) { case 1: insbeg(); break;

case 2:

insmid(); break;

case 3:

insend(); break; default:

printf("Invalid insertion choice\n"); break;

}

break; case 2:

printf("1. Begin 2. Middle 3. End "); printf("Enter your deletion choice: "); scanf("%d", &ch1);

switch (ch1) { case 1:

delbeg(); break;

case 2:

delmid(); break;

case 3:

delend(); break; default:

printf("Invalid deletion choice\n"); break;

}

break; case 3:

display(); break;

case 4:

exit(0); default:

printf("Invalid choice\n");

}

}

return 0;

}

void insbeg() {

Node \*temp = (Node \*)malloc(sizeof(Node)); int ele;

printf("Enter the element: "); scanf("%d", &ele);

temp->info = ele; temp->next = start; start = temp;

}

void insmid() {

Node \*temp = (Node \*)malloc(sizeof(Node)); int ele, pos, i;

printf("Enter the element: "); scanf("%d", &ele); printf("Enter the position: "); scanf("%d", &pos);

temp->info = ele;

if (pos == 1) {

temp->next = start; start = temp;

return;

}

Node \*ptr = start;

for (i = 1; i < pos - 1 && ptr != NULL; i++) { ptr = ptr->next;

}

if (ptr == NULL) {

printf("Position out of range\n"); free(temp);

return;

}

temp->next = ptr->next; ptr->next = temp;

}

void insend() {

Node \*temp = (Node \*)malloc(sizeof(Node)); int ele;

printf("Enter the element: "); scanf("%d", &ele);

temp->info = ele;

temp->next = NULL;

if (start == NULL) { start = temp; return;

}

Node \*ptr = start;

while (ptr->next != NULL) { ptr = ptr->next;

}

ptr->next = temp;

}

void delbeg() {

if (start == NULL) { printf("Underflow\n"); return;

}

Node \*ptr = start; start = start->next; free(ptr);

}

void delmid() { int pos, i;

if (start == NULL) { printf("Underflow\n"); return;

}

printf("Enter the position to delete: "); scanf("%d", &pos);

if (pos == 1) { delbeg(); return;

}

Node \*ptr = start; Node \*temp = NULL;

for (i = 1; i < pos && ptr != NULL; i++) { temp = ptr;

ptr = ptr->next;

}

if (ptr == NULL) {

printf("Position out of range\n"); return;

}

temp->next = ptr->next; free(ptr);

}

void delend() {

if (start == NULL) { printf("Underflow\n"); return;

}

if (start->next == NULL) { free(start);

start = NULL; return;

}

Node \*ptr = start; Node \*temp = NULL;

while (ptr->next != NULL) { temp = ptr;

ptr = ptr->next;

}

temp->next = NULL; free(ptr);

}

void display() {

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

}

Node \*ptr = start; printf("List elements: "); while (ptr != NULL) {

printf("%d ", ptr->info); ptr = ptr->next;

}

printf("\n");

}

Output :-

