**Que1: Queue implementation ( insertion, deletion,traversal)**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

int queue[MAX\_SIZE];

int front = -1;

int rear = -1;

void enqueue(int value) {

    if (front == 0 && rear == MAX\_SIZE - 1) {

        printf("Queue is full.\n");

        return;

    }

    if (front == -1) {

        front = 0;

    }

    for (int i = rear; i >= front; i--) {

        queue[i + 1] = queue[i];

    }

    queue[front] = value;

    rear++;

}

int dequeue() {

    if (front == -1 || front > rear) {

        printf("Queue is empty.\n");

        return -1;

    }

    int value = queue[rear];

    rear--;

    if (rear < front) {

        front = rear = -1;

    }

    return value;

}

void traverse() {

    if (front == -1) {

        printf("Queue is empty.\n");

        return;

    }

    printf("Queue elements:\n");

    for (int i = front; i <= rear; i++) {

        printf("%d ", queue[i]);

    }

    printf("\n");

}

int main() {

    int choice, value;

    while (1) {

        printf("1. Enqueue\n2. Dequeue\n3. Traverse\n4. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Enter value to enqueue: ");

                scanf("%d", &value);

                enqueue(value);

                break;

            case 2:

                value = dequeue();

                if (value != -1) {

                    printf("Dequeued value: %d\n", value);

                }

                break;

            case 3:

                traverse();

                break;

            case 4:

                exit(0);

            default:

                printf("Invalid choice.\n");

        }

    }

    return 0;

}

**Output**

1. Enqueue

2. Dequeue

3. Traverse

4. Exit

Enter your choice: 1

Enter value to enqueue: 100

1. Enqueue

2. Dequeue

3. Traverse

4. Exit

Enter your choice: 1

Enter value to enqueue: 200

1. Enqueue

2. Dequeue

3. Traverse

4. Exit

Enter your choice: 1

Enter value to enqueue: 300

1. Enqueue

2. Dequeue

3. Traverse

4. Exit

Enter your choice: 2

Dequeued value: 100

1. Enqueue

2. Dequeue

3. Traverse

4. Exit

Enter your choice: 3

Queue elements:

300 200

1. Enqueue

2. Dequeue

3. Traverse

4. Exit

**Que: Implementation of array ( insertion, deletion, searching, traversal, updation)**

#include <stdio.h>

#define MAX\_SIZE 100

int main() {

    int n = 0, arr[MAX\_SIZE], choice, element, index;

    while (1) {

        printf("1. Insert\n2. Delete\n3. Search\n4. Traverse\n5. Update\n6. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                if (n < MAX\_SIZE) {

                    printf("Enter element to insert: ");

                    scanf("%d", &element);

                    arr[n++] = element;

                } else {

                    printf("Array is full.\n");

                }

                break;

            case 2:

                printf("Enter index to delete: ");

                scanf("%d", &index);

                if (index >= 0 && index < n) {

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

                        arr[i] = arr[i + 1];

                    }

                    n--;

                } else {

                    printf("Invalid index.\n");

                }

                break;

            case 3:

                printf("Enter element to search: ");

                scanf("%d", &element);

                index = -1;

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

                    if (arr[i] == element) {

                        index = i;

                        break;

                    }

                }

                if (index != -1) {

                    printf("Element found at index %d.\n", index);

                } else {

                    printf("Element not found.\n");

                }

                break;

            case 4:

                printf("Array elements:\n");

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

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

                }

                printf("\n");

                break;

            case 5:

                printf("Enter index to update: ");

                scanf("%d", &index);

                if (index >= 0 && index < n) {

                    printf("Enter new value: ");

                    scanf("%d", &element);

                    arr[index] = element;

                } else {

                    printf("Invalid index.\n");

                }

                break;

            case 6:

                return 0;

            default:

                printf("Invalid choice.\n");

        }

    }

}

**OUTPUT:**

1. Insert

2. Delete

3. Search

4. Traverse

5. Update

6. Exit

Enter your choice: 1

Enter element to insert: 15

1. Insert

2. Delete

3. Search

4. Traverse

5. Update

6. Exit

Enter your choice: 1

Enter element to insert: 28

1. Insert

2. Delete

3. Search

4. Traverse

5. Update

6. Exit

Enter your choice: 1

Enter element to insert: 99

1. Insert

2. Delete

3. Search

4. Traverse

5. Update

6. Exit

Enter your choice: 4

Array elements:

15 28 99

1. Insert

2. Delete

3. Search

4. Traverse

5. Update

6. Exit

Enter your choice: 2

Enter index to delete: 2

1. Insert

2. Delete

3. Search

4. Traverse

5. Update

6. Exit

Enter your choice: 4

Array elements:

15 28

1. Insert

2. Delete

3. Search

4. Traverse

5. Update

6. Exit

Enter your choice: 3

Enter element to search: 28

Element found at index 1.

**Que:** **Stack implementation (insertion, deletion, traversal)**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

int stack[MAX\_SIZE];

int top = -1;

void push(int value) {

    if (top == MAX\_SIZE - 1) {

        printf("Stack is full.\n");

        return;

    }

    for (int i = top; i >= 0; i--) {

        stack[i + 1] = stack[i];

    }

    stack[0] = value;

    top++;

}

int pop() {

    if (top == -1) {

        printf("Stack is empty.\n");

        return -1;

    }

    int value = stack[0];

    for (int i = 0; i < top; i++) {

        stack[i] = stack[i + 1];

    }

    top--;

    return value;

}

void traverse() {

    if (top == -1) {

        printf("Stack is empty.\n");

        return;

    }

    printf("Stack elements:\n");

    for (int i = 0; i <= top; i++) {

        printf("%d ", stack[i]);

    }

    printf("\n");

}

int main() {

    int choice, value;

    while (1) {

        printf("1. Push\n2. Pop\n3. Traverse\n4. 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:

                value = pop();

                if (value != -1) {

                    printf("Popped value: %d\n", value);

                }

                break;

            case 3:

                traverse();

                break;

            case 4:

                exit(0);

            default:

                printf("Invalid choice.\n");

        }

    }

    return 0;

}

**Output**

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 1

Enter value to push: 10

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 1

Enter value to push: 25

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 1

Enter value to push: 35

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 3

Stack elements:

35 25 10

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 2

Popped value: 35

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 3

Stack elements:

25 10

1. Push

2. Pop

3. Traverse

4. Exit

**Que: Add element before and after a key element**

#include <stdio.h>

#define MAX\_SIZE 100

int main() {

    int n, arr[MAX\_SIZE], key, before, after;

    printf("Enter the number of elements (max %d): ", MAX\_SIZE);

    scanf("%d", &n);

    if (n > MAX\_SIZE) {

        printf("Error: Number of elements exceeds maximum size.\n");

        return 1;

    }

    printf("Enter the elements:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    printf("Enter the key element: ");

    scanf("%d", &key);

    printf("Enter the element to add before: ");

    scanf("%d", &before);

    printf("Enter the element to add after: ");

    scanf("%d", &after);

    int index = -1;

    for (int i = 0; i < n; i++) {

        if (arr[i] == key) {

            index = i;

            break;

        }}

    if (index != -1) {

        if (n + 2 > MAX\_SIZE) {

            printf("Error: Not enough space to add new elements.\n");

            return 1;

        }

        for (int i = n - 1; i >= index; i--) {

            arr[i + 2] = arr[i];

        }

        arr[index + 1] = after;

        arr[index-1] = before;

        n += 1;

    } else {

        printf("Key element not found in the array.\n");

    }

    printf("Array after adding elements:\n");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

    return 0;

}

**OUTPUT:**

Enter the number of elements (max 100): 5

Enter the elements:

10 20 30 40 50

Enter the key element: 30

Enter the element to add before: 111

Enter the element to add after: 222

Array after adding elements:

10 20 111 30 222 40 50

**Que: Bubble sort**

#include <stdio.h>

void bubbleSort(int arr[], int n) {

    for (int i = 0; i < n - 1; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

            }

        }

    }

}

int main() {

    int n;

    printf("Enter the number of elements in the array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements of the array:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    bubbleSort(arr, n);

    printf("Sorted array using Bubble Sort:\n");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

    return 0;

}

**OUTPUT:**

Enter the number of elements in the array: 5

Enter the elements of the array:

34 12 89 23 70

Sorted array using Bubble Sort:

12 23 34 70 89

**Que: Add element before and after a key element**

#include <stdio.h>

int main() {

    int n, key;

    printf("Enter the number of elements in the array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements of the array:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    printf("Enter the key : ");

    scanf("%d", &key);

    int minGreater = -1;

    int maxSmaller = -1;

    for (int i = 0; i < n; i++) {

        if (arr[i] > key) {

            if (minGreater == -1 || arr[i] < minGreater) {

                minGreater = arr[i];

            }

        }

        if (arr[i] < key) {

            if (maxSmaller == -1 || arr[i] > maxSmaller) {

                maxSmaller = arr[i];

            }

        }

    }

    if (minGreater == -1) {

        printf("No greater element  found.\n");

    } else {

        printf("The smallest element greater than %d is: %d\n", key, minGreater);

    }

    if (maxSmaller == -1) {

        printf("No smaller  element  found.\n");

    } else {

        printf("The largest element smaller than %d is: %d\n", key, maxSmaller);

    }

    return 0;

}

**OUTPUT:**

Enter the number of elements in the array: 6

Enter the elements of the array:

20 45 10 60 35 50

Enter the key : 35

The smallest element greater than 35 is: 45

The largest element smaller than 35 is: 20

**Que: Linkedlist Implementation.**

#include <stdio.h>

#include <stdlib.h>

struct Node {

    int data;

    struct Node\* next;

};

struct Node\* createNode(int data) {

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

    newNode->data = data;

    newNode->next = NULL;

    return newNode;

}

void insertAtStart(struct Node\*\* head, int data) {

    struct Node\* newNode = createNode(data);

    newNode->next = \*head;

    \*head = newNode;

}

void insertAtEnd(struct Node\*\* head, int data) {

    struct Node\* newNode = createNode(data);

    if (\*head == NULL) {

        \*head = newNode;

        return;  }

    struct Node\* temp = \*head;

    while (temp->next != NULL) {

        temp = temp->next; }

    temp->next = newNode;

}

void insertAtPosition(struct Node\*\* head, int data, int position) {

    if (position == 0) {

        insertAtStart(head, data);

        return;

    }

    struct Node\* newNode = createNode(data);

    struct Node\* temp = \*head;

    for (int i = 0; temp != NULL && i < position - 1; i++) {

        temp = temp->next;

    }

    if (temp == NULL) {

        printf("Position is out of bounds. Inserting at the end instead.\n");

        insertAtEnd(head, data);

        return;

    }

    newNode->next = temp->next;

    temp->next = newNode;

}

void displayList(struct Node\* head) {

    struct Node\* temp = head;

    while (temp != NULL) {

        printf("%d -> ", temp->data);

        temp = temp->next;

    }

    printf("NULL\n");

}

int main() {

    struct Node\* head = NULL;

    int choice, data, position;

    while (1) {

        printf("Choose an operation:\n");

        printf("1. Insert at Start\n");

        printf("2. Insert at End\n");

        printf("3. Insert at Position\n");

        printf("4. Display List\n");

        printf("5. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Enter data to insert at start: ");

                scanf("%d", &data);

                insertAtStart(&head, data);

                break;

            case 2:

                printf("Enter data to insert at end: ");

                scanf("%d", &data);

                insertAtEnd(&head, data);

                break;

            case 3:

                printf("Enter data to insert: ");

                scanf("%d", &data);

                printf("Enter position to insert at: ");

                scanf("%d", &position);

                insertAtPosition(&head, data, position);

                break;

            case 4:

                printf("Linked List: ");

                displayList(head);

                break;

            case 5:

                exit(0);

            default:

                printf("Invalid choice. Please try again.\n");

        }  }

    return 0;

}

**OUTPUT:**

Choose an operation:

1. Insert at Start

2. Insert at End

3. Insert at Position

4. Display List

5. Exit

Enter your choice: 1

Enter data to insert at start: 21

Choose an operation:

1. Insert at Start

2. Insert at End

3. Insert at Position

4. Display List

5. Exit

Enter your choice: 1

Enter data to insert at start: 30

Choose an operation:

1. Insert at Start

2. Insert at End

3. Insert at Position

4. Display List

5. Exit

Enter your choice: 2

Enter data to insert at end: 99

Choose an operation:

1. Insert at Start

2. Insert at End

3. Insert at Position

4. Display List

5. Exit

Enter your choice: 3

Enter data to insert: 50

Enter position to insert at: 2

Choose an operation:

1. Insert at Start

2. Insert at End

3. Insert at Position

4. Display List

5. Exit

Enter your choice: 4

Linked List: 30 -> 21 -> 50 -> 99 -> NULL

Choose an operation:

1. Insert at Start

2. Insert at End

3. Insert at Position

4. Display List

5. Exit

Enter your choice: 4

Linked List: 30 -> 21 -> 50 -> 99 -> NULL

**Que:** **Second and third repeating element in array**

#include <stdio.h>

#define MAX\_SIZE 100

int main() {

    int n, arr[MAX\_SIZE];

    int count[MAX\_SIZE] = {0};

    int second = -1, third = -1;

    printf("Enter the number of elements in the array: ");

    scanf("%d", &n);

    printf("Enter the elements: \n");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    for (int i = 0; i < n; i++) {

        count[arr[i]]++;

    }

    for (int i = 0; i < MAX\_SIZE; i++) {

        if (count[i] > 1) {

            if (second == -1) {

                second = i;

            } else if (third == -1) {

                third = i;

                break;

            }

        }

    }

    if (second != -1) {

        printf("Second repeating element: %d\n", second);

    } else {

        printf("No second repeating element\n");

    }

    if (third != -1) {

        printf("Third repeating element: %d\n", third);

    } else {

        printf("No third repeating element\n");

    }

    return 0;

}

**OUTPUT:**

Enter the number of elements in the array: 5

Enter the elements:

22 45 33 22 55

Second repeating element: 22

Third repeating element: 45

**Que:** **Queue implementation ( insertion, deletion,traversal)**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

int queue[MAX\_SIZE];

int front = -1;

int rear = -1;

void enqueue(int value) {

    if ((rear + 1) % MAX\_SIZE == front) {

        printf("Queue is full.\n");

        return;

    }

    if (front == -1) {

        front = 0;

    }

    rear = (rear + 1) % MAX\_SIZE;

    queue[rear] = value;

}

int dequeue() {

    if (front == -1) {

        printf("Queue is empty.\n");

        return -1;

    }

    int value = queue[front];

    if (front == rear) {

        front = rear = -1;

    } else {

        front = (front + 1) % MAX\_SIZE;

    }

    return value;

}

void traverse() {

    if (front == -1) {

        printf("Queue is empty.\n");

        return;

    }

    printf("Queue elements:\n");

    for (int i = front; i != rear; i = (i + 1) % MAX\_SIZE) {

        printf("%d ", queue[i]);

    }

    printf("%d\n", queue[rear]);

}

int main() {

    int choice, value;

    while (1) {

        printf("1. Enqueue\n2. Dequeue\n3. Traverse\n4. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Enter value to enqueue: ");

                scanf("%d", &value);

                enqueue(value);

                break;

            case 2:

                value = dequeue();

                if (value != -1) {

                    printf("Dequeued value: %d\n", value);

                }

                break;

            case 3:

                traverse();

                break;

            case 4:

                exit(0);

            default:

                printf("Invalid choice.\n");

        }

    }

    return 0;

}

**1. Enqueue**

**2. Dequeue**

**3. Traverse**

**4. Exit**

**Enter your choice: 1**

**Enter value to enqueue: 200**

**1. Enqueue**

**2. Dequeue**

**3. Traverse**

**4. Exit**

**Enter your choice: 1**

**Enter value to enqueue: 400**

**1. Enqueue**

**2. Dequeue**

**3. Traverse**

**4. Exit**

**Enter your choice: 1**

**Enter value to enqueue: 600**

**1. Enqueue**

**2. Dequeue**

**3. Traverse**

**4. Exit**

**Enter your choice: 3**

**Queue elements:**

**200 400 600**

**1. Enqueue**

**2. Dequeue**

**3. Traverse**

**4. Exit**

**Enter your choice: 2**

**Dequeued value: 200**

**1. Enqueue**

**2. Dequeue**

**3. Traverse**

**4. Exit**

**Enter your choice: 3**

**Queue elements:**

**400 600**

**Que: Reverse each row of a matrix**

#include <stdio.h>

#define MAX\_SIZE 100

int main() {

    int n, m, matrix[MAX\_SIZE][MAX\_SIZE];

    printf("Enter the number of rows and columns: ");

    scanf("%d %d", &n, &m);

    printf("Enter the elements of the matrix:\n");

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < m; j++) {

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

        }

    }

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < m / 2; j++) {

            int temp = matrix[i][j];

            matrix[i][j] = matrix[i][m - j - 1];

            matrix[i][m - j - 1] = temp;

        }

    }

    printf("Matrix after reversing each row:\n");

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < m; j++) {

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

        }

        printf("\n");

    }

    return 0;

}

**OUTPUT:**

Enter the number of rows and columns: 4 4

Enter the elements of the matrix:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Matrix after reversing each row:

4 3 2 1

8 7 6 5

12 11 10 9

16 15 14 13

**Que: Reverse of spiral matrix**

#include <stdio.h>

#define MAX\_SIZE 100

void fillSpiralMatrix(int n, int matrix[MAX\_SIZE][MAX\_SIZE]) {

    int left = 0, right = n - 1, top = 0, bottom = n - 1;

    int num = 1;

    while (left <= right && top <= bottom) {

        for (int i = left; i <= right; i++) {

            matrix[top][i] = num++;

        }

        top++;

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

            matrix[i][right] = num++;

        }

        right--;

        if (top <= bottom) {

            for (int i = right; i >= left; i--) {

                matrix[bottom][i] = num++;

            }

            bottom--;

        }

        if (left <= right) {

            for (int i = bottom; i >= top; i--) {

                matrix[i][left] = num++;

            }

            left++;

        }

    }

}

void printMatrix(int n, int matrix[MAX\_SIZE][MAX\_SIZE]) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

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

        }

        printf("\n");

    }

}

void printReversedMatrix(int n, int matrix[MAX\_SIZE][MAX\_SIZE]) {

    for (int i = n - 1; i >= 0; i--) {

        for (int j = n - 1; j >= 0; j--) {

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

        }

        printf("\n");

    }

}

int main() {

    int n;

    int matrix[MAX\_SIZE][MAX\_SIZE];

    printf("Enter the size of the matrix (n x n): ");

    scanf("%d", &n);

    if (n <= 0 || n > MAX\_SIZE) {

        printf("Error: Size must be between 1 and 100.\n");

        return 1;

    }

    fillSpiralMatrix(n, matrix);

    printf("Spiral Matrix:\n");

    printMatrix(n, matrix);

    printf("Reversed Spiral Matrix:\n");

    printReversedMatrix(n, matrix);

    return 0;

}

**OUTPUT:**

Enter the size of the matrix (n x n): 4

Spiral Matrix:

1 2 3 4

12 13 14 5

11 16 15 6

10 9 8 7

Reversed Spiral Matrix:

7 8 9 10

6 5 4 11

3 2 1 12

13 14 15 16

**Que: Rotate matrix by 90 degree**

#include <stdio.h>

#define MAX\_SIZE 100

void rotateMatrix(int matrix[MAX\_SIZE][MAX\_SIZE], int n) {

    for (int layer = 0; layer < n / 2; layer++) {

        int first = layer;

        int last = n - layer - 1;

        for (int i = first; i < last; i++) {

            int offset = i - first;

            int top = matrix[first][i];

            matrix[first][i] = matrix[last - offset][first];

            matrix[last - offset][first] = matrix[last][last - offset];

            matrix[last][last - offset] = matrix[i][last];

            matrix[i][last] = top;

        }

    }}

void printMatrix(int matrix[MAX\_SIZE][MAX\_SIZE], int n) {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

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

        }

        printf("\n");

    }

}

int main() {

    int n;

    int matrix[MAX\_SIZE][MAX\_SIZE];

    printf("Enter the size of the matrix (n x n): ");

    scanf("%d", &n);

    printf("Enter the elements of the matrix:\n");

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

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

        }

    }

    rotateMatrix(matrix, n);

    printf("Matrix after 90 degree rotation:\n");

    printMatrix(matrix, n);

    return 0;

}

**OUTPUT:**

Enter the size of the matrix (n x n): 3

Enter the elements of the matrix:

12 34 56

78 90 12

34 56 78

Matrix after 90 degree rotation:

34 78 12

56 90 34

78 12 56

**Que:** **Rotate array with k times**

#include <stdio.h>

void rotateArray(int arr[], int n, int k) {

    k = k % n;

    int temp[n];

    for (int i = 0; i < n; i++) {

        temp[(i + k) % n] = arr[i];

    }

    for (int i = 0; i < n; i++) {

        arr[i] = temp[i];

    }

}

int main() {

    int n, k;

    printf("Enter the number of elements in the array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter the elements of the array:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    printf("Enter the number of rotations: ");

    scanf("%d", &k);

    rotateArray(arr, n, k);

    printf("Array after rotation:\n");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

    return 0;

**OUTPUT:**

Enter the number of elements in the array: 6

Enter the elements of the array:

12 45 23 67 89 34

Enter the number of rotations: 2

Array after rotation:

23 67 89 34 12 45

**Que: Search an element and insert another element in its position**

#include <stdio.h>

#define MAX\_SIZE 100

int main() {

    int n, arr[MAX\_SIZE], key, newElement;

    printf("Enter the number of elements: ");

    scanf("%d", &n);

    printf("Enter the elements:\n");

    for (int i = 0; i < n; i++) {

        scanf("%d", &arr[i]);

    }

    printf("Enter the key element to search: ");

    scanf("%d", &key);

    printf("Enter the new element to insert: ");

    scanf("%d", &newElement);

    int index = -1;

    for (int i = 0; i < n; i++) {

        if (arr[i] == key) {

            index = i;

            break;

        }

    }

    if (index != -1) {

        for (int i = n; i > index; i--) {

            arr[i + 1] = arr[i];

        }

        arr[index] = newElement;

        n++;

    }

    printf("Array after insertion:\n");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

    return 0;

}

**OUTPUT:**

Enter the number of elements: 4

Enter the elements:

100 200 300 400

Enter the key element to search: 200

Enter the new element to insert: 350

Array after insertion:

100 200 350 300 400

**Que: Shifting of one element in spiral matrix**

#include <stdio.h>

#define MAX\_SIZE 100

int main() {

    int n, matrix[MAX\_SIZE][MAX\_SIZE];

    printf("Enter the size of the matrix (n x n): ");

    scanf("%d", &n);

    if (n <= 0 || n > MAX\_SIZE) {

        printf("Error: Size must be between 1 and %d.\n", MAX\_SIZE);

        return 1;

    }

    printf("Enter the elements of the matrix:\n");

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

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

        }

    }

    int lastElement = matrix[0][0];

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

            if (i == 0 && j == 0) {

                continue;

            }

            int temp = matrix[i][j];

            matrix[i][j] = lastElement;

            lastElement = temp;

        }

    }

    matrix[0][0] = lastElement;

    printf("Matrix after shifting:\n");

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

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

        }

        printf("\n");

    }

    return 0;

}

**OUTPUT:**

Enter the size of the matrix (n x n): 4

Enter the elements of the matrix:

10 20 30 40

50 60 70 80

90 100 110 120

130 140 150 160

Matrix after shifting:

40 10 20 30

80 50 60 70

120 90 100 110

160 130 140 150

**Que: Spiral matrix**

#include <stdio.h>

void generateSpiralMatrix(int n) {

    int matrix[n][n];

    int left = 0, right = n - 1, top = 0, bottom = n - 1;

    int num = 1;

    while (left <= right && top <= bottom) {

        for (int i = left; i <= right; i++) {

            matrix[top][i] = num++;

        }

        top++;

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

            matrix[i][right] = num++;

        }

        right--;

        if (top <= bottom) {

            for (int i = right; i >= left; i--) {

                matrix[bottom][i] = num++;

            }

            bottom--;

        }        if (left <= right) {

            for (int i = bottom; i >= top; i--) {

                matrix[i][left] = num++;

            }

            left++;

        }

    }

    printf("Spiral Matrix:\n");

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < n; j++) {

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

        }

        printf("\n");

    }

}

int main() {

    int n;

    printf("Enter the size of the matrix (n x n): ");

    scanf("%d", &n);

    generateSpiralMatrix(n);

    return 0;

}

**OUTPUT:**

Enter the size of the matrix (n x n): 3

Spiral Matrix:

1 2 3 4

12 13 14 5

11 16 15 6

10 9 8 7

**Que:** **Insertion from 0th index and deletion from last index**

#include <stdio.h>

#include <stdlib.h>

#define MAX\_SIZE 100

int stack[MAX\_SIZE];

int top = -1;

void push(int value) {

    if (top == MAX\_SIZE - 1) {

        printf("Stack is full.\n");

        return;

    }

    stack[++top] = value;

}

int pop() {

    if (top == -1) {

        printf("Stack is empty.\n");

        return -1; // Return -1 or some error value

    }

    return stack[top--];

}

void traverse() {

    if (top == -1) {

        printf("Stack is empty.\n");

        return;

    }

    printf("Stack elements:\n");

    for (int i = 0; i <= top; i++) {

        printf("%d ", stack[i]);

    }

    printf("\n");

}

int main() {

    int choice, value;

    while (1) {

        printf("1. Push\n2. Pop\n3. Traverse\n4. 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:

                value = pop();

                if (value != -1) {

                    printf("Popped value: %d\n", value);

                }

                break;

            case 3:

                traverse();

                break;

            case 4:

                exit(0);

            default:

                printf("Invalid choice.\n");

        }

    }

    return 0;

}

**OUTPUT:**

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 1

Enter value to push: 20

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 1

Enter value to push: 45

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 1

Enter value to push: 10

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 2

Popped value: 10

1. Push

2. Pop

3. Traverse

4. Exit

Enter your choice: 3

Stack elements:

20 45

**Que: Transpose of a matrix without using extra array**

#include <stdio.h>

int main() {

    printf("Enter the size: ");

    int N;

    scanf("%d", &N);

    int matrix[N][N];

    printf("Enter the elements of matrix:\n");

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < N; j++) {

            printf("Element : ");

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

        }

    }

    printf("Original Mat\n");

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < N; j++) {

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

        }

        printf("\n");

    }

    for (int i = 0; i < N; i++) {

        for (int j = i + 1; j < N; j++) {

            int temp = matrix[i][j];

            matrix[i][j] = matrix[j][i];

            matrix[j][i] = temp;

        }

    }

    printf("Transposed Mat:\n");

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < N; j++) {

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

        }

        printf("\n");

    }

    return 0;

}

**OUTPUT:**

Enter the size: 3

Enter the elements of matrix:

10 20 30

40 50 60

70 80 90

Original Mat:

10 20 30

40 50 60

70 80 90

Transposed Mat:

10 40 70

20 50 80

30 60 90