1

```c
#include <stdio.h>
int main() {
    int i, j;
    for(i=1; i<=4; i++) {
        for(j=1; j<=i; j++)
            printf("%d", j);
        printf("\n");
    }
    return 0;
}
```

```c
#include <stdio.h>
int main() {
    int i, j, k, n = 4;

    // Upper Half
    for(i=1; i<=n; i++) {
        for(j=i; j<n; j++)
            printf(" ");
        for(k=1; k<=i; k++)
            printf("%d ", i);
        printf("\n");
    }

    // Lower Half
    for(i=n-1; i>=1; i--) {
        for(j=n; j>i; j--)
            printf(" ");
        for(k=1; k<=i; k++)
            printf("%d ", i);
        printf("\n");
    }
    return 0;
}
```

2

```c
#include <stdio.h>
int main() {
    int i, j;
    for(i=1; i<=4; i++) {
        for(j=1; j<=i; j++)
            printf("*");
        printf("\n");
    }
    return 0;
}
```

```c
#include <stdio.h>
int main() {
    int i, j;
    char ch;
    for(i=1; i<=4; i++) {
        for(j=i; j<4; j++)
            printf(" ");
        for(ch='A'; ch<'A'+i; ch++)
            printf("%c", ch);
        for(ch='A'+i-2; ch>='A'; ch--)
            printf("%c", ch);
        printf("\n");
    }
    return 0;
}
```

3

```c
#include <stdio.h>

struct Student {
    int roll;
    char name[30];
    float marks;
};

int main() {
    struct Student s[50];
    int n, i;

    printf("Enter number of students: ");
    scanf("%d", &n);

    // Create Records
    printf("\nEnter details:\n");
    for(i = 0; i < n; i++) {
        printf("\nStudent %d\n", i+1);
        printf("Enter Roll No: ");
        scanf("%d", &s[i].roll);
        printf("Enter Name: ");
        scanf("%s", s[i].name);
        printf("Enter Marks: ");
        scanf("%f", &s[i].marks);
    }

    // Display Records
    printf("\n---- Student Database ----\n");
    printf("Roll\tName\tMarks\n");
    for(i = 0; i < n; i++) {
        printf("%d\t%s\t%.2f\n", s[i].roll, s[i].name, s[i].marks);
    }

    return 0;
}
```

4

```c
 #include <stdio.h>

int main() {
    int arr[50], n, i, key, found = 0;

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

    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("Enter element to search: ");
    scanf("%d", &key);

    for(i = 0; i < n; i++) {
        if(arr[i] == key) {
            printf("Element found at position %d\n", i + 1);
            found = 1;
            break;
        }
    }

    if(!found)
        printf("Element not found\n");

    return 0;
}


#include <stdio.h>

int main() {
    int arr[50], n, i, key, low, high, mid;

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

    printf("Enter elements in sorted order:\n");
    for(i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("Enter element to search: ");
    scanf("%d", &key);
```

```
    low = 0;
    high = n - 1;

    while(low <= high) {
        mid = (low + high) / 2;

        if(arr[mid] == key) {
            printf("Element found at position %d\n", mid + 1);
            return 0;
        }
        else if(arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }

    printf("Element not found\n");
    return 0;
}
```

5

```c
#include <stdio.h>

// User-defined function for Bubble Sort
void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for(i = 0; i < n - 1; i++) {
        for(j = 0; j < n - i - 1; j++) {
            if(arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int arr[50], n, i;

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

    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    bubbleSort(arr, n);

    printf("\nSorted array:\n");
    for(i = 0; i < n; i++)
        printf("%d ", arr[i]);

    return 0;
}
```

6

```c
#include <stdio.h>

// Function to perform selection sort
void selectionSort(int arr[], int n) {
    int i, j, minIndex, temp;
    for (i = 0; i < n - 1; i++) {
        minIndex = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex])
                minIndex = j;
        }
        // Swap the found minimum element with the first element
        temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
}

// Function to display array
void display(int arr[], int n) {
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
int main() {
    int arr[50], n, i;

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

    printf("Enter array elements:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("\nOriginal Array: ");
    display(arr, n);

    selectionSort(arr, n);

    printf("Sorted Array (Selection Sort): ");
    display(arr, n);

    return 0;
}
```

7

```c
#include <stdio.h>

// Function for Insertion Sort
void insertionSort(int arr[], int n) {
    int i, j, key;
    for(i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        // Move elements greater than key one position ahead
        while(j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

// Function to display array
void display(int arr[], int n) {
    int i;
    for(i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[50], n, i;
    printf("Enter number of elements: ");
    scanf("%d", &n);

    printf("Enter array elements:\n");
    for(i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("\nOriginal Array: ");
    display(arr, n);

    insertionSort(arr, n);

    printf("Sorted Array (Insertion Sort): ");
    display(arr, n);

    return 0;
}
```

8

```c
#include <stdio.h>
#define SIZE 5

int stack[SIZE], top = -1;

void push(int val) {
    if (top == SIZE - 1)
        printf("Overflow\n");
    else
        stack[++top] = val;
}

void pop() {
    if (top == -1)
        printf("Underflow\n");
    else
        printf("Popped: %d\n", stack[top--]);
}

void display() {
    if (top == -1)
        printf("Empty\n");
    else {
        for (int i = top; i >= 0; i--)
            printf("%d ", stack[i]);
        printf("\n");
    }
}

int main() {
    int ch, val;
    while (1) {
        printf("\n1.Push  2.Pop  3.Display  4.Exit: ");
        scanf("%d", &ch);

        if (ch == 1) {
            printf("Enter value: ");
            scanf("%d", &val);
            push(val);
        }
        else if (ch == 2)
            pop();
        else if (ch == 3)
            display();
        else if (ch == 4)
```

```
            break;
        else
            printf("Invalid choice\n");
    }
    return 0;
}
```

9

```
#include <stdio.h>
#define SIZE 5

int queue[SIZE], front = -1, rear = -1;

void enqueue(int val) {
    if (rear == SIZE - 1)
        printf("Overflow\n");
    else {
        if (front == -1) front = 0;
        queue[++rear] = val;
    }
}

void dequeue() {
    if (front == -1 || front > rear)
        printf("Underflow\n");
    else
        printf("Deleted: %d\n", queue[front++]);
}

void display() {
    if (front == -1 || front > rear)
        printf("Empty\n");
    else {
        for (int i = front; i <= rear; i++)
            printf("%d ", queue[i]);
        printf("\n");
    }
}

int main() {
    int ch, val;
    while (1) {
        printf("\n1.Enqueue 2.Dequeue 3.Display 4.Exit: ");
        scanf("%d", &ch);

        if (ch == 1) {
```

```
            printf("Enter value: ");
            scanf("%d", &val);
            enqueue(val);
        }
        else if (ch == 2)
            dequeue();
        else if (ch == 3)
            display();
        else if (ch == 4)
            break;
        else
            printf("Invalid choice\n");
    }
    return 0;
}
```

10

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

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

struct Node* head = NULL;

void insertFront(int val) {
    struct Node* newNode = malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->next = head;
    head = newNode;
}

void insertEnd(int val) {
    struct Node* newNode = malloc(sizeof(struct Node));
    newNode->data = val;
    newNode->next = NULL;

    if (head == NULL)
        head = newNode;
    else {
        struct Node* temp = head;
        while (temp->next != NULL)
            temp = temp->next;
```

```
        temp->next = newNode;
    }
}

void deleteFront() {
    if (head == NULL)
        printf("List Empty\n");
    else {
        struct Node* temp = head;
        head = head->next;
        free(temp);
    }
}

void deleteEnd() {
    if (head == NULL)
        printf("List Empty\n");
    else if (head->next == NULL) {
        free(head);
        head = NULL;
    } else {
        struct Node* temp = head;
        while (temp->next->next != NULL)
            temp = temp->next;
        free(temp->next);
        temp->next = NULL;
    }
}

void display() {
    struct Node* temp = head;
    if (temp == NULL)
        printf("List Empty\n");
    else {
        while (temp != NULL) {
            printf("%d ", temp->data);
            temp = temp->next;
        }
        printf("\n");
    }
}

int main() {
    int ch, val;
    while (1) {
        printf("\n1.InsertFront 2.InsertEnd 3.DelFront 4.DelEnd
```

```
5.Display 6.Exit: ");
        scanf("%d", &ch);

        if (ch == 1) { scanf("%d", &val); insertFront(val); }
        else if (ch == 2) { scanf("%d", &val); insertEnd(val); }
        else if (ch == 3) deleteFront();
        else if (ch == 4) deleteEnd();
        else if (ch == 5) display();
        else if (ch == 6) break;
        else printf("Invalid choice\n");
    }
    return 0;
}
```