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

#define MAX 5

//STATIC FUNCTION DECLARATIONS
void staticMenu();
void readArrayStatic(int arr[], int *n);
void displayStatic(int arr[], int n);
void insertPosStatic(int arr[], int *n, int pos, int val);
void deletePosStatic(int arr[], int *n, int pos);
void insertKeyStatic(int arr[], int *n, int key, int val);
void deleteKeyStatic(int arr[], int *n, int key);
void insertOrderStatic(int arr[], int *n, int val);
int searchKeyStatic(int arr[], int n, int key);
int searchPosStatic(int arr[], int n, int pos);
void reverseStatic(int arr[], int n);

//DYNAMIC FUNCTION DECLARATIONS
void dynamicMenu();
int* readArrayDynamic(int *arr, int *n);
int* insertPosDynamic(int *arr, int *n, int pos, int val);
int* insertKeyDynamic(int *arr, int *n, int key, int val);
int* insertOrderDynamic(int *arr, int *n, int val);
void deletePosDynamic(int *arr, int *n, int pos);
void deleteKeyDynamic(int *arr, int *n, int key);
int searchKeyDynamic(int *arr, int n, int key);
int searchPosDynamic(int *arr, int n, int pos);
void reverseDynamic(int *arr, int n);
int* ensureCapacity(int *arr, int *n, int *capacity);

int main()
{
    int choice;
    do
    {
        printf("\n========== MAIN MENU ==========\n");
        printf("1. Static Array Operations (Fixed size)\n");
        printf("2. Dynamic Array Operations (malloc + realloc)\n");
        printf("0. Exit\n");
        printf("================================\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                staticMenu();
                break;
            case 2:
                dynamicMenu();
                break;
            case 0:
                printf("Exiting program...\n");
                break;
            default:
                printf("Invalid choice!\n");
        }
    } while (choice != 0);
    return 0;
}

void staticMenu()
{
    int arr[MAX], n = 0, choice, pos, key, val, result;
    do
    {
        printf("\n------ STATIC ARRAY MENU ------\n");
        printf("1. Read Array\n2. Display\n3. Insert by Position\n4. Delete by Position\n");
        printf("5. Insert by Key\n6. Delete by Key\n7. Insert in Order\n");
        printf("8. Search by Key\n9. Search by Position\n10. Reverse\n0. Back\n");
        printf("--------------------------------\nEnter choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1: readArrayStatic(arr, &n); break;
            case 2: displayStatic(arr, n); break;
            case 3:
                printf("Enter pos & value: ");
                scanf("%d %d", &pos, &val);
                insertPosStatic(arr, &n, pos, val);
                break;
            case 4:
                printf("Enter position: ");
                scanf("%d", &pos);
                deletePosStatic(arr, &n, pos);
                break;
            case 5:
                printf("Enter key & value: ");
                scanf("%d %d", &key, &val);
                insertKeyStatic(arr, &n, key, val);
                break;
            case 6:
                printf("Enter key to delete: ");
                scanf("%d", &key);
                deleteKeyStatic(arr, &n, key);
                break;
            case 7:
                printf("Enter value: ");
                scanf("%d", &val);
                insertOrderStatic(arr, &n, val);
                break;
            case 8:
                printf("Enter key: ");
                scanf("%d", &key);
                result = searchKeyStatic(arr, n, key);
                (result != -1) ? printf("Found at pos %d\n", result) : printf("Not found\n");
                break;
            case 9:
                printf("Enter position: ");
                scanf("%d", &pos);
                result = searchPosStatic(arr, n, pos);
                if (result != -1) printf("Value = %d\n", result);
                break;
            case 10:
                reverseStatic(arr, n);
                printf("Reversed.\n");
                break;
            case 0:
                return;
        }
    } while (choice != 0);
}

void readArrayStatic(int arr[], int *n)
{
    int size, val;
    *n = 0;
    printf("Enter size (max %d): ", MAX);
    scanf("%d", &size);
    for (int i = 0; i < size; i++)
    {
        printf("Enter value: ");
        scanf("%d", &val);
        insertOrderStatic(arr, n, val);
    }
}

void displayStatic(int arr[], int n)
{
    if (n == 0)
    {
        printf("Empty Array.\n");
        return;
    }
    for (int i = 0; i < n; i++)
    printf("%d ", arr[i]);
    printf("\n");
}

void insertPosStatic(int arr[], int *n, int pos, int val)
{
    if (*n >= MAX)
    {
        printf("Array Full.\n");
        return;
    }
    if (pos < 1 || pos > *n + 1)
    {
        printf("Invalid Position.\n");
        return;
    }
    for (int i = *n; i >= pos; i--)
        arr[i] = arr[i - 1];
    arr[pos - 1] = val;
    (*n)++;
}

void deletePosStatic(int arr[], int *n, int pos)
{
    if (*n == 0 || pos < 1 || pos > *n)
    {
        printf("Invalid Position.\n");
        return;
    }
    for (int i = pos - 1; i < *n - 1; i++)
        arr[i] = arr[i + 1];
    (*n)--;
}

void insertKeyStatic(int arr[], int *n, int key, int val)
{
    int i, found = 0;
    for (i = 0; i < *n; i++)
        if (arr[i] == key)
        { found = 1;
            break;
        }
    if (!found)
    {
        printf("Key not found.\n");
        return;
    }
    if (*n >= MAX)
    {
        printf("Full.\n");
        return;
    }

    for (int j = *n; j > i + 1; j--)
        arr[j] = arr[j - 1];
    arr[i + 1] = val;
    (*n)++;
}

void deleteKeyStatic(int arr[], int *n, int key)
{
    int i, found = 0;

    for (i = 0; i < *n; i++)
        if (arr[i] == key)
        { found = 1;
            break;
        }
    if (!found)
    {
        printf("Key not found.\n");
        return;
    }
    for (int j = i; j < *n - 1; j++)
        arr[j] = arr[j + 1];
    (*n)--;
}

void insertOrderStatic(int arr[], int *n, int val)
{
    if (*n >= MAX)
    {
        printf("Array Full.\n");
        return;
    }

    int i = *n - 1;
    while (i >= 0 && arr[i] > val)
    {
        arr[i + 1] = arr[i];
        i--;
    }
    arr[i + 1] = val;
    (*n)++;
}

int searchKeyStatic(int arr[], int n, int key)
{
    for (int i = 0; i < n; i++)
        if (arr[i] == key)
        return i + 1;
    return -1;
}

int searchPosStatic(int arr[], int n, int pos)
{
    if (pos < 1 || pos > n)
    return -1;
    return arr[pos - 1];
}

void reverseStatic(int arr[], int n)
```

```
{
    int temp;
    for (int i = 0; i < n / 2; i++)
    {
        temp = arr[i];
        arr[i] = arr[n - 1 - i];
        arr[n - 1 - i] = temp;
    }
}

void dynamicMenu()
{
    int n = 0, choice, pos, key, val, result;
    int capacity = 5;

    int *arr = (int *)malloc(capacity * sizeof(int));

    do {
        printf("\n------ DYNAMIC ARRAY MENU ------\n");
        printf("1. Read Array\n2. Display\n3. Insert by Position\n4. Delete by Position\n");
        printf("5. Insert by Key\n6. Delete by Key\n7. Insert in Order\n");
        printf("8. Search by Key\n9. Search by Position\n10. Reverse\n0. Back\n");
        printf("--------------------------------\nEnter choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
            case 1: arr = readArrayDynamic(arr, &n); break;
            case 2: displayStatic(arr, n); break;
            case 3:
                printf("Enter pos & value: ");
                scanf("%d %d", &pos, &val);
                arr = insertPosDynamic(arr, &n, pos, val);
                break;
            case 4:
                printf("Enter pos: ");
                scanf("%d", &pos);
                deletePosDynamic(arr, &n, pos);
                break;
            case 5:
                printf("Enter key & value: ");
                scanf("%d %d", &key, &val);
                arr = insertKeyDynamic(arr, &n, key, val);
                break;
            case 6:
                printf("Enter key: ");
                scanf("%d", &key);
                deleteKeyDynamic(arr, &n, key);
                break;
            case 7:
                printf("Enter value: ");
                scanf("%d", &val);
                arr = insertOrderDynamic(arr, &n, val);
                break;
            case 8:
                printf("Enter key: ");
                scanf("%d", &key);
                result = searchKeyDynamic(arr, n, key);
                (result != -1) ? printf("Found at pos %d\n", result) : printf("Not found\n");
                break;
            case 9:
                printf("Enter position: ");
                scanf("%d", &pos);
                result = searchPosDynamic(arr, n, pos);
                if (result != -1)
                printf("Value = %d\n", result);
                break;
            case 10:
                reverseDynamic(arr, n);
                printf("Reversed.\n");
                break;
            case 0:
                free(arr);
                return;
        }
    } while (choice != 0);
}

int* ensureCapacity(int *arr, int *n, int *capacity)
{
    if (*n >= *capacity)
    {
        *capacity *= 2;
        arr = realloc(arr, (*capacity) * sizeof(int));
    }
    return arr;
```

```
}

int* readArrayDynamic(int *arr, int *n)
{
    int size, val, capacity = 5;
    *n = 0;
    printf("Enter size: ");
    scanf("%d", &size);
    for (int i = 0; i < size; i++)
    {
        printf("Enter value: ");
        scanf("%d", &val);
        arr = insertOrderDynamic(arr, n, val);
    }
    return arr;
}

int* insertPosDynamic(int *arr, int *n, int pos, int val)
{
    static int capacity = 5;
    arr = ensureCapacity(arr, n, &capacity);

    if (pos < 1 || pos > *n + 1)
    return arr;

    for (int i = *n; i >= pos; i--)
        arr[i] = arr[i - 1];

    arr[pos - 1] = val;
    (*n)++;
    return arr;
}

void deletePosDynamic(int *arr, int *n, int pos)
{
    if (pos < 1 || pos > *n)
    return;
    for (int i = pos - 1; i < *n - 1; i++)
        arr[i] = arr[i + 1];
    (*n)--;
}

int* insertKeyDynamic(int *arr, int *n, int key, int val)
```

```
{
    static int capacity = 5;
    int pos = -1;
    for (int i = 0; i < *n; i++)
        if (arr[i] == key)
        {
            pos = i;
            break;
        }

    if (pos == -1)
        return arr;
    arr = ensureCapacity(arr, n, &capacity);
    for (int i = *n; i > pos + 1; i--)
        arr[i] = arr[i - 1];
    arr[pos + 1] = val;
    (*n)++;
    return arr;
}

void deleteKeyDynamic(int *arr, int *n, int key)
{
    int pos = -1;
    for (int i = 0; i < *n; i++)
        if (arr[i] == key)
        {
            pos = i;
            break;
        }

    if (pos == -1)
    return;
    for (int i = pos; i < *n - 1; i++)
        arr[i] = arr[i + 1];
    (*n)--;
}

int* insertOrderDynamic(int *arr, int *n, int val)
{
    static int capacity = 5;
    arr = ensureCapacity(arr, n, &capacity);
    int i = *n - 1;
    while (i >= 0 && arr[i] > val)
    {
        arr[i + 1] = arr[i];
        i--;
    }
    arr[i + 1] = val;
    (*n)++;
    return arr;
}

int searchKeyDynamic(int *arr, int n, int key)
{
    for (int i = 0; i < n; i++)
        if (arr[i] == key)
        return i + 1;
    return -1;
}

int searchPosDynamic(int *arr, int n, int pos)
{
    if (pos < 1 || pos > n)
    return -1;
    return arr[pos - 1];
}

void reverseDynamic(int *arr, int n)
{
    int temp;
    for (int i = 0; i < n / 2; i++)
    {
        temp = arr[i];
        arr[i] = arr[n - 1 - i];
        arr[n - 1 - i] = temp;
    }
}
```