Program 5: Menu-driven program for Stack (Array Implementation)

```c
#include <stdio.h>
#define SIZE 50

int stack[SIZE], top = -1;

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

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

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

int main() {
    int choice, item;
    do {
        printf("\n--- STACK MENU ---\n");
        printf("1. PUSH\n2. POP\n3. DISPLAY\n4. EXIT\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: printf("Enter element: ");
                    scanf("%d", &item);
                    push(item);
                    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;
}
```

Program 6: Palindrome Check using Stack

```c
#include <stdio.h>
#include <string.h>
```

```
#define SIZE 50

char stack[SIZE];
int top = -1;

void push(char c) {
    stack[++top] = c;
}
char pop() {
    return stack[top--];
}

int main() {
    char str[50];
    int i, flag = 1;

    printf("Enter a string: ");
    scanf("%s", str);

    for (i = 0; i < strlen(str); i++)
        push(str[i]);

    for (i = 0; i < strlen(str); i++) {
        if (str[i] != pop()) {
            flag = 0;
            break;
        }
    }

    if (flag)
        printf("Palindrome\n");
    else
        printf("Not Palindrome\n");

    return 0;
}
```

Program 8: Reverse a String using Stack

```
#include <stdio.h>
#include <string.h>
#define SIZE 50

char stack[SIZE];
int top = -1;

void push(char c) {
    stack[++top] = c;
}
char pop() {
    return stack[top--];
}

int main() {
    char str[50];
    int i;

    printf("Enter a string: ");
    scanf("%s", str);
    for (i = 0; i < strlen(str); i++)
        push(str[i]);

    printf("Reversed string: ");
    for (i = 0; i < strlen(str); i++)
        printf("%c", pop());

    return 0;
}
```

Program 9: Fibonacci Numbers using Stack

```
#include <stdio.h>
#define SIZE 50

int stack[SIZE];
int top = -1;

void push(int n) {
    stack[++top] = n;
}
int pop() {
    return stack[top--];
}

int main() {
    int n, i, a = 0, b = 1, c;

    printf("Enter N: ");
    scanf("%d", &n);

    push(a);
    push(b);

    for (i = 2; i < n; i++) {
        a = pop();
        b = stack[top];
        push(a);
        c = a + b;
        push(c);
    }

    printf("Fibonacci Series:\n");
    for (i = 0; i <= top; i++)
        printf("%d ", stack[i]);

    return 0;
}
```

Program 10: Factorial using Stack

```
#include <stdio.h>
#define SIZE 50

int stack[SIZE];
int top = -1;
```

```
void push(int n) {
    stack[++top] = n;
}
int pop() {
    return stack[top--];
}

int main() {
    int n, i;
    long fact = 1;

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

    for (i = 1; i <= n; i++)
        push(i);

    while (top != -1)
        fact *= pop();

    printf("Factorial = %ld\n", fact);
    return 0;
}
```

Program 11: Balanced Brackets using Stack

```
#include <stdio.h>
#include <string.h>
#define SIZE 50

char stack[SIZE];
int top = -1;

void push(char c) { stack[++top] = c; }
char pop() { return stack[top--]; }

int main() {
    char exp[50], c;
    int i, flag = 1;

    printf("Enter expression: ");
    scanf("%s", exp);

    for (i = 0; i < strlen(exp); i++) {
        if (exp[i] == '(' || exp[i] == '{' || exp[i] == '[')
            push(exp[i]);
        else if (exp[i] == ')' || exp[i] == '}' || exp[i] == ']') {
            if (top == -1) { flag = 0; break; }
            c = pop();
            if ((exp[i] == ')' && c != '(') ||
                (exp[i] == '}' && c != '{') ||
                (exp[i] == ']' && c != '[')) {
                flag = 0; break;
            }
        }
    }
    if (top != -1) flag = 0;
    if (flag)
        printf("Brackets are Balanced\n");
    else
        printf("Brackets are NOT Balanced\n");

    return 0;
}
```

Program 12: Bubble Sort (Numbers in Array)

```
#include <stdio.h>

void bubbleSort(int a[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1]) {
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
}

int main() {
    int a[20], n, i;
    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    bubbleSort(a, n);

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

    return 0;
}
```

Program 13: Bubble Sort (Strings in Character Matrix)

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

void bubbleSort(char str[20][20], int n) {
    char temp[20];
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (strcmp(str[j], str[j + 1]) > 0) {
                strcpy(temp, str[j]);
                strcpy(str[j], str[j + 1]);
                strcpy(str[j + 1], temp);
```

```c
            }
        }
    }
}

int main() {
    char str[20][20];
    int n, i;

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

    printf("Enter strings:\n");
    for (i = 0; i < n; i++)
        scanf("%s", str[i]);

    bubbleSort(str, n);

    printf("Sorted strings:\n");
    for (i = 0; i < n; i++)
        printf("%s\n", str[i]);

    return 0;
}
```

Program 14: Selection Sort (Numbers in Array)

```c
#include <stdio.h>

void selectionSort(int a[], int n) {
    int i, j, min, temp;
    for (i = 0; i < n - 1; i++) {
        min = i;
        for (j = i + 1; j < n; j++) {
            if (a[j] < a[min])
                min = j;
        }
        temp = a[i];
        a[i] = a[min];
        a[min] = temp;
    }
}

int main() {
    int a[20], n, i;
    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    selectionSort(a, n);

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

    return 0;
}
```

Program 15: Linear Search using Function

```
#include <stdio.h>

int fnSearchNum(int n, int a[20], int searchNum) {
    int i;
    for (i = 0; i < n; i++) {
        if (a[i] == searchNum)
            return 1;   // found
    }
    return -1;  // not found
}

int main() {
    int a[20], n, i, searchNum, result;

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

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

    printf("Enter number to search: ");
    scanf("%d", &searchNum);

    result = fnSearchNum(n, a, searchNum);

    if (result == 1)
        printf("Number found\n");
    else
        printf("Number not found\n");

    return 0;
}
```

You said:

Binary Search Tree Menu Program in C

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

struct node {
    int data;
    struct node *left, *right;
};

struct node* createNode(int value) {
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
```

```c
    return newNode;
}

struct node* insert(struct node* root, int value) {
    if (root == NULL)
        return createNode(value);
    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
        root->right = insert(root->right, value);
    return root;
}

void inorder(struct node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}

void preorder(struct node* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void postorder(struct node* root) {
    if (root != NULL) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}

struct node* search(struct node* root, int key) {
    if (root == NULL || root->data == key)
        return root;
    if (key < root->data)
        return search(root->left, key);
    else
        return search(root->right, key);
}

void displayLeafNodes(struct node* root) {
    if (root != NULL) {
        if (root->left == NULL && root->right == NULL)
            printf("%d ", root->data);
        displayLeafNodes(root->left);
        displayLeafNodes(root->right);
    }
}

int isLeaf(struct node* root, int key) {
    struct node* temp = search(root, key);
    if (temp != NULL && temp->left == NULL && temp->right == NULL)
        return 1;
    return 0;
}
```

```
int main() {
    struct node* root = NULL;
    int choice, num;

    do {
        printf("\n--- BINARY SEARCH TREE MENU ---\n");
        printf("1. Insert\n");
        printf("2. Inorder Traversal\n");
        printf("3. Postorder Traversal\n");
        printf("4. Preorder Traversal\n");
        printf("5. Search for a Number\n");
        printf("6. Display Leaf Nodes\n");
        printf("7. Search & Check if Leaf Node\n");
        printf("8. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch(choice) {
            case 1: printf("Enter number to insert: ");
                    scanf("%d", &num);
                    root = insert(root, num);
                    break;

            case 2: printf("Inorder: ");
                    inorder(root);
                    printf("\n");
                    break;

            case 3: printf("Postorder: ");
                    postorder(root);
                    printf("\n");
                    break;

            case 4: printf("Preorder: ");
                    preorder(root);
                    printf("\n");
                    break;

            case 5: printf("Enter number to search: ");
                    scanf("%d", &num);
                    if (search(root, num) != NULL)
                        printf("Number found\n");
                    else
                        printf("Number not found\n");
                    break;

            case 6: printf("Leaf Nodes: ");
                    displayLeafNodes(root);
                    printf("\n");
                    break;

            case 7: printf("Enter number to check leaf: ");
                    scanf("%d", &num);
                    if (isLeaf(root, num))
                        printf("%d is a Leaf Node\n", num);
                    else
                        printf("%d is NOT a Leaf Node\n", num);
                    break;

            case 8: printf("Exiting...\n");
```

```
                    break;

                default: printf("Invalid Choice\n");
            }
        } while(choice != 8);

        return 0;
}
```

You said:

Program 14: Selection Sort

```
#include <stdio.h>

void selectionSort(int a[], int n) {
    int i, j, min, temp;
    for (i = 0; i < n - 1; i++) {
        min = i;
        for (j = i + 1; j < n; j++) {
            if (a[j] < a[min])
                min = j;
        }
        temp = a[i];
        a[i] = a[min];
        a[min] = temp;
    }
}
```

```
int main() {
    int a[20], n, i;
    printf("Enter number of elements: ");
    scanf("%d", &n);

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

    selectionSort(a, n);

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

    return 0;
}
```

Program 15: Linear Search (with Function)

```
#include <stdio.h>

int fnSearchNum(int n, int a[20], int searchNum) {
    int i;
    for (i = 0; i < n; i++) {
        if (a[i] == searchNum)
            return 1;   // found
    }
    return -1;          // not found
}

int main() {
    int a[20], n, i, searchNum, result;

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

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

    printf("Enter number to search: ");
    scanf("%d", &searchNum);

    result = fnSearchNum(n, a, searchNum);

    if (result == 1)
        printf("Number found\n");
    else
        printf("Number not found\n");

    return 0;
}
```

Program 16: Binary Search (with Function)

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

```c
int fnBinarySearch(int n, int a[20], int searchNum) {
    int low = 0, high = n - 1, mid;
    while (low <= high) {
        mid = (low + high) / 2;
        if (a[mid] == searchNum)
            return 1;   // found
        else if (a[mid] < searchNum)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;          // not found
}

int main() {
    int a[20], n, i, searchNum, result;

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

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

    printf("Enter number to search: ");
    scanf("%d", &searchNum);

    result = fnBinarySearch(n, a, searchNum);

    if (result == 1)
        printf("Number found\n");
    else
        printf("Number not found\n");

    return 0;
}
```

Program 17: Insertion Sort

```c
#include <stdio.h>

void insertionSort(int a[], int n) {
    int i, j, key;
    for (i = 1; i < n; i++) {
        key = a[i];
        j = i - 1;
        while (j >= 0 && a[j] > key) {
            a[j + 1] = a[j];
            j--;
        }
        a[j + 1] = key;
    }
}

int main() {
    int a[20], n, i;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter elements: ");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    insertionSort(a, n);

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

    return 0;
}
```