C Programming Practice Programs

1. Stack and its operations

```
#include <stdio.h>
#define SIZE 100

int stack[SIZE], top = -1;

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

int pop() {
    if (top == -1) {
        printf("Stack Underflow\n");
        return -1;
    } else
        return stack[top--];
}

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

int peek() {
    if (top == -1) {
        printf("Stack is empty\n");
        return -1;
    } else
        return stack[top];
}

int search(int key) {
    for (int i = top; i >= 0; i--) {
        if (stack[i] == key)
            return i;
    }
    return -1;
}

int main() {
    push(10);
    push(20);
    push(30);
    traverse();
    printf("Peek: %d\n", peek());
    printf("Search 20: %d\n", search(20));
```

```
    printf("Popped: %d\n", pop());
    traverse();
    return 0;
}
```

2. Infix to Postfix Expression Conversion

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

#define SIZE 100
char stack[SIZE];
int top = -1;

void push(char ch) {
    stack[++top] = ch;
}

char pop() {
    return stack[top--];
}

int precedence(char ch) {
    switch (ch) {
        case '^': return 3;
        case '*':
        case '/': return 2;
        case '+':
        case '-': return 1;
        default: return 0;
    }
}

void infixToPostfix(char* infix, char* postfix) {
    int k = 0;
    for (int i = 0; infix[i]; i++) {
        char ch = infix[i];
        if (isalnum(ch)) postfix[k++] = ch;
        else if (ch == '(') push(ch);
        else if (ch == ')') {
            while (top != -1 && stack[top] != '(')
                postfix[k++] = pop();
            pop(); // remove '('
        } else {
            while (top != -1 && precedence(stack[top]) >= precedence(ch))
                postfix[k++] = pop();
            push(ch);
        }
    }
    while (top != -1) postfix[k++] = pop();
    postfix[k] = '\0';
}

int main() {
    char infix[SIZE], postfix[SIZE];
    printf("Enter infix: ");
    gets(infix);
    infixToPostfix(infix, postfix);
    printf("Postfix: %s\n", postfix);
```

```
    return 0;
}
```