


```
    st->top++;
    st->S[st->top] = x;
}
}
```

```
// Function to pop an element from the stack
```

```
int pop(Stack *st) {
    int x = -1;
    if (st->top == -1) {
        cout << "Stack Underflow." << endl;
    } else {
        x = st->S[st->top];
        st->top--;
    }
    return x;
}
```

```
// Function to peek at an element at a given position
```

```
int peek(Stack st, int pos) {
    int x = -1;
    if (st.top - pos + 1 < 0) {
        cout << "Invalid position." << endl;
    } else {
        x = st.S[st.top - pos + 1];
    }
    return x;
}
```

```
// Function to check if stack is empty
```

```
int isEmpty(Stack st) {
    return st.top == -1;
```

```
}

// Function to check if stack is full

int isFull(Stack st) {
    return st.top == st.size - 1;
}

}

int main() {
    Stack st; // Create a stack object
    create(&st); // Initialize the stack
    push(&st, 10); // Push elements
    push(&st, 20);
    push(&st, 30);
    push(&st, 40);
    push(&st, 50);
    pop(&st); // Pop one element
    display(st); // Display stack
    cout << peek(st, 3) << endl; // Peek at position 3

    delete[] st.S; // Free allocated memory

    return 0;
}
```

- Enter size: 4
Stack Overflow.
30
20
10
10

```

void push(Stack* st, char x) {
    if (st->top == st->size - 1) {
        cout << "Stack Overflow! Cannot push '" << x << "'." << endl;
    } else {
        st->top++;
        st->S[st->top] = x;
        cout << "Pushed '" << x << "' to stack." << endl;
    }
}

char pop(Stack* st) {
    char x = '\0';
    if (st->top == -1) {
        cout << "Stack Underflow! Nothing to pop." << endl;
    } else {
        x = st->S[st->top];
        st->top--;
        cout << "Popped '" << x << "' from stack." << endl;
    }
    return x;
}

void display(Stack st) {
    if (st.top == -1) {
        cout << "Stack is empty." << endl;
        return;
    }
    cout << "Stack contents (top to bottom): ";
    for (int i = st.top; i >= 0; i--) {
        cout << st.S[i];
    }
    cout << endl;
}

int main() {
    Stack st;
    char string[] = "DataStructure";
    int n = strlen(string);

    create(&st, n);

    for (int i = 0; i < n; i++) {
        push(&st, string[i]);
    }

    display(st);

    delete[] st.S;

    return 0;
}

```

- Pushed 'D' to stack.
 - Pushed 'a' to stack.
 - Pushed 't' to stack.
 - Pushed 'a' to stack.
 - Pushed 's' to stack.
 - Pushed 't' to stack.
 - Pushed 'r' to stack.
 - Pushed 'u' to stack.
 - Pushed 'c' to stack.
 - Pushed 't' to stack.
 - Pushed 'u' to stack.
 - Pushed 'r' to stack.
 - Pushed 'e' to stack.

Stack contents (top to bottom): erutcurtSataD

```

42 void display(Stack st) {
43     for (int i = st.top; i >= 0; i--) {
44         cout << st.S[i];
45     }
46     cout << endl;
47 }
48
49 bool isMatchingPair(char opening, char closing) {
50     return (opening == '(' && closing == ')') ||
51            (opening == '{' && closing == '}') ||
52            (opening == '[' && closing == ']');
53 }
54
55 int main() {
56     Stack st;
57     char arr[] = "(({}[]))";
58     int n = strlen(arr);
59     create(&st, n);
60     for (int i = 0; i < n; i++) {
61         if (arr[i] == '(' || arr[i] == '{' || arr[i] == '[') {
62             push(&st, arr[i]);
63         } else if (arr[i] == ')' || arr[i] == '}' || arr[i] == ']') {
64             if (isEmpty(st)) {
65                 cout << "Parenthesis is unmatched" << endl;
66                 delete[] st.S;
67                 return 0;
68             }
69             if (!isMatchingPair(st.S[st.top], arr[i])) {
70                 cout << "Parenthesis is unmatched" << endl;
71                 delete[] st.S;
72                 return 0;
73             }
74             pop(&st);
75         }
76     }
77
78     if (isEmpty(st)) {
79         cout << "Parenthesis is matched" << endl;
80     } else {
81         cout << "Parenthesis is unmatched" << endl;
82     }
83
84     delete[] st.S;
85     return 0;

```

Parenthesis is unmatched

4>>>>>>>>>>>>>>>>>..

```
49 int isOperand(char x) {
50     if (x == '+' || x == '-' || x == '*' || x == '/') {
51         return 0;
52     } else {
53         return 1;
54     }
55 }
56
57 int precedence(char x) {
58     if (x == '+' || x == '-') {
59         return 1;
60     }
61     if (x == '*' || x == '/') {
62         return 2;
63     }
64     return 0;
65 }
66
67 int main() {
68     Stack st;
69     create(&st, 100);
70     char infix[] = "a+b*c+d-e/f";
71     char postfix[100] = {0};
72     int i = 0, j = 0;
73
74     while (infix[i] != '\0') {
75         if (isOperand(infix[i])) {
76             postfix[j++] = infix[i++];
77         } else {
78             while (!isEmpty(st) && precedence(st.S[st.top]) >= precedence(infix[i])) {
79                 postfix[j++] = pop(&st);
80             }
81             push(&st, infix[i++]);
82         }
83     }
84
85     while (!isEmpty(st)) {
86         postfix[j++] = pop(&st);
87     }
88     postfix[j] = '\0';
89
90     cout << "Infix: " << infix << endl;
91     cout << "Postfix: " << postfix << endl;
92
93     delete[] st.S;
94     return 0;
95 }
```

Infix: $a+b*c+d-e/f$
Postfix: abc*+d+ef/-

```

48 int isoperand(char x) {
49     if (x == '+' || x == '-' || x == '*' || x == '/') {
50         return 0;
51     } else {
52         return 1;
53     }
54 }
55 }
56
57 int precedence(char x) {
58     if (x == '+' || x == '-') {
59         return 1;
60     }
61     if (x == '*' || x == '/') {
62         return 2;
63     }
64     return 0;
65 }
66
67 int main() {
68     Stack st;
69     create(&st, 100);
70     char infix[] = "3+7*5+4-5/2";
71     char postfix[100] = {0};
72     int i = 0, j = 0;
73
74     while (infix[i] != '\0') {

while (infix[i] != '\0') {
    if (isOperand(infix[i])) {
        postfix[j++] = infix[i++];
    } else {
        while (!isEmpty(st) && precedence(st.S[st.top]) >= precedence(infix[i])) {
            postfix[j++] = pop(&st);
        }
        push(&st, infix[i++]);
    }
}

while (!isEmpty(st)) {
    postfix[j++] = pop(&st);
}
postfix[j] = '\0';

i = 0;
int x1, x2, r = 0;
for (i = 0; postfix[i] != '\0'; i++) {
    if (isOperand(postfix[i])) {
        push(&st, postfix[i] - '0');
    } else {
        x2 = pop(&st);
        x1 = pop(&st);
        switch (postfix[i]) {
            case '+': r = x1 + x2; break;
            case '-': r = x1 - x2; break;
            case '*': r = x1 * x2; break;
            case '/': r = x1 / x2; break;
        }
        push(&st, r);
    }
}

cout << (int)st.S[st.top] << endl;

delete[] st.S;
return 0;
}

```

