```
// C program to evaluate value of a postfix expression
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Stack type
struct Stack {
      int top;
      unsigned capacity;
      int* array;
};

// Stack Operations
struct Stack* createStack(unsigned capacity)
{
      struct Stack* stack
            = (struct Stack*)malloc(sizeof(struct Stack));

      if (!stack)
            return NULL;

      stack->top = -1;
      stack->capacity = capacity;
      stack->array
            = (int*)malloc(stack->capacity * sizeof(int));

      if (!stack->array)
            return NULL;

      return stack;
}

int isEmpty(struct Stack* stack)
{
      return stack->top == -1;
}

char peek(struct Stack* stack)
{
      return stack->array[stack->top];
}

char pop(struct Stack* stack)
{
      if (!isEmpty(stack))
            return stack->array[stack->top--];
      return '$';
}

void push(struct Stack* stack, char op)
{
      stack->array[++stack->top] = op;
}
```

```
// The main function that returns value
// of a given postfix expression
int evaluatePostfix(char* exp)
{
      // Create a stack of capacity equal to expression size
      struct Stack* stack = createStack(strlen(exp));
      int i;

      // See if stack was created successfully
      if (!stack)
            return -1;

      // Scan all characters one by one
      for (i = 0; exp[i]; ++i) {

            // If the scanned character is an operand
            // (number here), push it to the stack.
            if (isdigit(exp[i]))
                  push(stack, exp[i] - '0');

            // If the scanned character is an operator,
            // pop two elements from stack apply the operator
            else {
                  int val1 = pop(stack);
                  int val2 = pop(stack);
                  switch (exp[i]) {
                  case '+':
                        push(stack, val2 + val1);
                        break;
                  case '-':
                        push(stack, val2 - val1);
                        break;
                  case '*':
                        push(stack, val2 * val1);
                        break;
                  case '/':
                        push(stack, val2 / val1);
                        break;
                  }
            }
      }
      return pop(stack);
}

// Driver code
int main()
{
      char exp[] = "231*+9-";

      // Function call
      printf("postfix evaluation: %d", evaluatePostfix(exp));
      return 0;
}
```