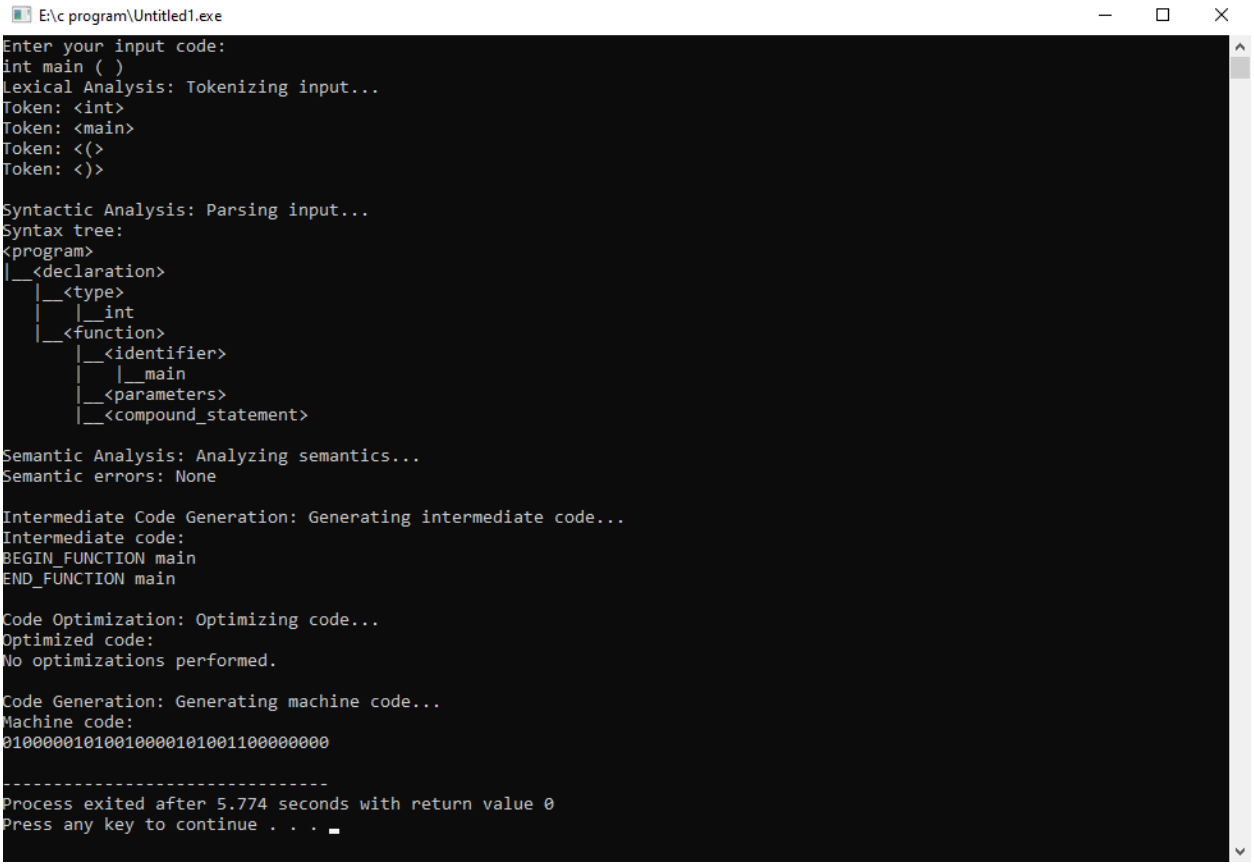
15. Write a C program to implement the back end of the compiler.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>


int isDelimiter(char ch) {

    return ch == ' ' || ch == '(' || ch == ')' || ch == '{' || ch == '}' || ch == '\n';

}


void lexicalAnalysis(char *input) {

    printf("Lexical Analysis: Tokenizing input...\n");

    char *token = strtok(input, " ");

    while (token != NULL) {

        printf("Token: <%s>\n", token);

        token = strtok(NULL, " ");

    }

}


void syntacticAnalysis(char *input) {

    printf("\nSyntactic Analysis: Parsing input...\n");

    printf("Syntax tree:\n");

    printf("<program>\n");
```

```
    printf("|__<declaration>\n");
    printf("   |__<type>\n");
    printf("   |   |__int\n");
    printf("   |__<function>\n");
    printf("       |__<identifier>\n");
    printf("       |   |__main\n");
    printf("       |__<parameters>\n");
    printf("       |__<compound_statement>\n");
}

void semanticAnalysis(char *input) {
    printf("\nSemantic Analysis: Analyzing semantics...\n");
    printf("Semantic errors: None\n");
}

void intermediateCodeGeneration(char *input) {
    printf("\nIntermediate Code Generation: Generating intermediate code...\n");
    printf("Intermediate code:\n");
    printf("BEGIN_FUNCTION main\n");
    printf("END_FUNCTION main\n");
}

void codeOptimization(char *input) {
```

```
    printf("\nCode Optimization: Optimizing code...\n");
    printf("Optimized code:\n");
    printf("No optimizations performed.\n");
}

void codeGeneration(char *input) {
    printf("\nCode Generation: Generating machine code...\n");
    printf("Machine code:\n");
    printf("01000001010010000101001100000000\n"); // Example machine code
}

int main() {
    char input[100];
    printf("Enter your input code:\n");
    fgets(input, sizeof(input), stdin);
    input[strcspn(input, "\n")] = '\0'; // Remove newline character
    lexicalAnalysis(input);
    syntacticAnalysis(input);
    semanticAnalysis(input);
    intermediateCodeGeneration(input);
    codeOptimization(input);
    codeGeneration(input);
    return 0;
```

}


E:\c program\Untitled1.exe
Enter your input code:
int main ( )
Lexical Analysis: Tokenizing input...
Token: <int>
Token: <main>
Token: <(>
Token: <)>
Syntactic Analysis: Parsing input...
Syntax tree:
<program>
|__<declaration>
|__<type>
|__int
|__<function>
|__<identifier>
|__main
|__<parameters>
|__<compound_statement>
Semantic Analysis: Analyzing semantics...
Semantic errors: None
Intermediate Code Generation: Generating intermediate code...
Intermediate code:
BEGIN_FUNCTION main
END_FUNCTION main
Code Optimization: Optimizing code...
Optimized code:
No optimizations performed.
Code Generation: Generating machine code...
Machine code:
01000001010010000101001100000000
Process exited after 5.774 seconds with return value 0
Press any key to continue . . .