9. Implementation of Shift Reduce Parsing Algorithm

```
#include <stdio.h>
#include <string.h>
char input[50], stack[50] = "$", temp[10];
int i = 0;
void push(char *s) {
   strcat(stack, s);
}
void reduce() {int len = strlen(stack);
   if (stack[len - 2] == 'i' && stack[len - 1] == 'd') {stack[len - 2] = 'E'; stack[len - 1] = '\0';
} else if (stack[len - 1] == 'E' && stack[len - 3] == 'E') {stack[len - 3] = 'E'; stack[len - 2] = '\0';
int main() {printf("LR PARSING\nENTER THE EXPRESSION\n");scanf("%s", input);strcat(input,
"$");printf("$\n");
   while (input[i] != '\0') {
      sprintf(temp, "%c", input[i++]);push(temp);printf("%s\n", stack);
      while (1) {int before = strlen(stack);reduce();if (before == strlen(stack)) break; printf("%s\n",
stack);}}
      return 0;}
```

10. Construction of Operator Precedence Parse Table

```
#include <stdio.h>
#include <string.h>
#define N 4
char symbols[N] = {'i', '+', '*', '$'};
char table[N][N]; // precedence table
int get_index(char c) {
   for (int i = 0; i < N; i++)if (symbols[i] == c) return i;return -1;}
void print_table() {
   printf("\n**** OPERATOR PRECEDENCE TABLE ****\n\n\t");
   for (int i = 0; i < N; i++) printf("%c\t", symbols[i]);
   printf("\n");
   for (int i = 0; i < N; i++) {
      printf("%c\t", symbols[i]);for (int j = 0; j < N; j++) { if (table[i][j] == 'a') printf("acc\t");
         else if (table[i][j]) printf("%c\t", table[i][j]); else printf("e\t");} printf("\n");}}
int main() {
   char input[50], stack[50] = "$"; int top = 0; for (int i = 0; i < N; i++)
   for (int j = 0; j < N; j++) { printf("Enter the value for %c %c: ", symbols[i], symbols[j]);
   scanf(" %c", &table[i][j]);}
   print_table(); printf("\nEnter the input string: "); scanf("%s", input);
   strcat(input, "$");printf("\n%-25s%-25s%-20s\n", "STACK", "INPUT STRING", "ACTION");
   int ip = 0;
   while (1) {
      char a = stack[top];char b = input[ip];
      int row = get_index(a), col = get_index(b);
      printf("%-25s%-25s", stack, input + ip);
      if (table[row][col] == '<' || table[row][col] == '=') {stack[++top] = input[ip++];
         stack[top + 1] = '\0';printf("Shift %c\n", stack[top]);
      } else if (table[row][col] == '>') {stack[top--] = '\0';printf("Reduce\n");
      } else if (table[row][col] == 'a') {printf("Accepted\n");break;
      } else {printf("Rejected\n");break;}} return 0;}
```

11. Implementation of Quadruples

```
#include <stdio.h>
#include <string.h>
int temp_count = 1;
void print_quad(const char *op, const char *arg1, const char *arg2, const char *res) {
   printf("%-4s %-4s %-4s %-4s\n", op, arg1, arg2, res);}
int main() {
   char expr[100], left[10], right[90]; char t1[10], t2[10], t3[10], t4[10], t5[10];
   printf("Enter a String : "); fgets(expr, sizeof(expr), stdin);
   expr[strcspn(expr, "\n")] = '\0';sscanf(expr, "%[^=]=%s", left, right);
   printf("\nop   a1   a2   res\n");  sprintf(t1, "t%d", temp_count++);
   print_quad(":=", "c", "", t1);      sprintf(t2, "t%d", temp_count++);
   print_quad("*", "b", t1, t2);       sprintf(t3, "t%d", temp_count++);
   print_quad(":=", "c", "", t3);      sprintf(t4, "t%d", temp_count++);
   print_quad("*", "b", t3, t4);       sprintf(t5, "t%d", temp_count++);
   print_quad("-", t2, t4, t5);        print_quad(":=", t5, "", left);
   return 0;}
```

13. Implementation of Intermediate Code Generation

```
#include <stdio.h>
#include <string.h>
int temp_count = 1;
void generate_code(FILE *out, const char *op, const char *arg1, const char *arg2, const char
*res) {
   fprintf(out, "%s=%s%s%s\n", res, arg1, op, arg2);
}
void process_expression(FILE *in, FILE *out) {
   char line[100], lhs[10], rhs[90]; char t[10]; fgets(line, sizeof(line), in);
   sscanf(line, "%[^=]=%s", lhs, rhs); char *token = strtok(rhs, " ");
   strcpy(t, "t1");
while (token != NULL) {
      if (strcmp(token, "+") == 0 || strcmp(token, "-") == 0) {
         char op[2]; strcpy(op, token); token = strtok(NULL, " ");
         char operand1[10], operand2[10]; strcpy(operand1, token); token = strtok(NULL, " ");
         strcpy(operand2, token);
         char temp[10];sprintf(temp, "t%d", temp_count++);generate_code(out, op, operand1,
         operand2, temp);strcpy(t, temp);
         } else { strcpy(t, token);}token = strtok(NULL, " ");}fprintf(out, "%s=%s\n", lhs, t);}
int main() { FILE *in, *out; in = fopen("sum.txt", "r");
   if (in == NULL) {printf("Error opening input file\n");return 1;}
   out = fopen("out.txt", "w");
   if (out == NULL) {printf("Error opening output file\n");return 1;}
   process_expression(in, out);
   fclose(in); fclose(out);
   printf("Intermediate code generated in out.txt\n"); return 0;}
```

14. Implementation of Code Generation

```
#include <stdio.h>
#include <string.h>
void main() {
   char icode[10][30], str[20], opr[10];
   int i = 0; printf("\nEnter intermediate code:\t");
   do {
      scanf("%s", icode[i]);
   } while (strcmp(icode[i++], "exit") != 0); printf("\nTarget Code Generation\n"); i = 0;
   do {
      strcpy(str, icode[i]);
      switch (str[3]) {
         case '+':
            strcpy(opr, "ADD");
            break;
         case '-':
            strcpy(opr, "SUB");
            break;
         case '*':
            strcpy(opr, "MUL");
            break;
         case '/':
            strcpy(opr, "DIV");
            break; }
      printf("\n\tMov %c, R%d", str[2], i); printf("\n\t%s %c, R%d, R%d", opr, str[0], i, i + 1);
      printf("\n\tMov R%d, %c", i + 1, str[4]);i++;} while (strcmp(icode[i], "exit") != 0);}
```

15. Implementation of Code Optimization Techniques

```
#include <stdio.h>
int main() {
   int n, choice, i, fact = 1;
   printf("Choose method to calculate:\n"); printf("for loop\n"); printf("do-while loop\n");
   printf("Enter(1 or 2): "); scanf("%d", &choice); printf("Enter an integer: ");scanf("%d", &n);
   if (n < 0) {
      printf("Factorial is not defined for negative numbers.\n");
      return 1;}
   switch (choice) {
      case 1:
         fact = 1; for (i = 1; i <= n; ++i) {
            fact *= i;} printf("The factorial value is: %d\n", fact);
         break;
      case 2:
         { int temp = n; fact = 1;
            do { fact *= temp; temp--;
            } while (temp > 0);
            printf("The factorial value is: %d\n", fact);}
         break;
      default:
         printf("Invalid choice.\n");
         break;}
   return 0;}
```