back end of compiler.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct {
    char result[20];
    char operand1[20];
    char operand2[20];
    char operator[5];
} ThreeAddressCode;
void generateAssemblyCode(ThreeAddressCode *code, int codeCount) {
    printf("Generated Assembly Code:\n");

    for (int i = 0; i < codeCount; i++) {
        printf("%s = %s %s %s\n", code[i].result, code[i].operand1, code[i].operator,
code[i].operand2);
    }
}
int main() {
    ThreeAddressCode intermediateCode[] = {
        {"t1", "a", "b", "+"},
        {"t2", "t1", "c", "*"},
        {"result", "t2", "d", "-"}
    };
    int codeCount = sizeof(intermediateCode) / sizeof(intermediateCode[0]);
    generateAssemblyCode(intermediateCode, codeCount);
            return 0;
}
```

counts characters,words,lines.c

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

int main() {
    char ch;
    int charCount = 0, wordCount = 0, lineCount = 0;
    int inWord = 0;
    printf("Enter text (Ctrl+D to end input):\n");

    while ((ch = getchar()) != EOF) {
        charCount++;
        if (ch == '\n') {
            lineCount++;
        }
```

```
        if (ch == ' ' || ch == '\t' || ch == '\n') {
            inWord = 0;
        } else if (!inWord) {
            inWord = 1;
            wordCount++;
        }
    }
    printf("\nAnalysis Results:\n");
    printf("Number of characters: %d\n", charCount);
    printf("Number of words: %d\n", wordCount);
    printf("Number of lines: %d\n", lineCount);

    return 0;
}
```

9.Satisfying the grammar or not .

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

int checkGrammar(const char *sentence) {
    // Implement your grammar rules
    // Return 1 if the sentence satisfies the grammar, 0 otherwise

    // Example: Check if the sentence starts with "The" and ends with a period.
    if (strncmp(sentence, "The", 3) == 0 && sentence[strlen(sentence) - 1] == '.') {
        return 1;
    } else {
        return 0;
    }
}

int main() {
    char sentence[100];

    printf("Enter a sentence: ");
    gets(sentence);

    if (checkGrammar(sentence)) {
        printf("The sentence satisfies the grammar.\n");
    } else {
        printf("The sentence does not satisfy the grammar.\n");
    }

    return 0;
```

```
}
8. Symbol Table Operations .c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int cnt=0;
struct symtab
{
        char label[20];
        int addr;
}
sy[50];
void insert();
int search(char *);
void display();
void modify();
int main()
{
int ch,val;
char lab[10];
do
{
        printf("\n1.insert\n2.display\n3.search\n4.modify\n5.exit\n");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1:
                        insert();
                         break;
                        case 2:
                                display();
                                break;
                case 3:
printf("enter the label");
                        scanf("%s",lab);
                        val=search(lab);
                        if(val==1)
                        printf("label is found");
                        else
                        printf("label is not found");
                break;
        case 4:
                        modify();
                break;
```

```
        case 5:
                        exit(0);
                        break;
                }
        }while(ch<5);
}
void insert()
{
int val;
        char lab[10];
        int symbol;
        printf("enter the label");
        scanf("%s",lab);
        val=search(lab);
        if(val==1)
        printf("duplicate symbol");
        else
        {
                strcpy(sy[cnt].label,lab);
                printf("enter the address");
                scanf("%d",&sy[cnt].addr);
                cnt++;
        }
}
int search(char *s)
{
        int flag=0,i; for(i=0;i<cnt;i++)
        {
                if(strcmp(sy[i].label,s)==0)
                flag=1;
        }
return flag;
}
void modify()
{
        int val,ad,i;
        char lab[10];
        printf("enter the labe:");
        scanf("%s",lab);
        val=search(lab);
        if(val==0)
        printf("no such symbol");
        else
        {
```

```
                printf("label is found \n");
                printf("enter the address");
                scanf("%d",&ad);
                for(i=0;i<cnt;i++)
                {
                        if(strcmp(sy[i].label,lab)==0)
                        sy[i].addr=ad;
                }
        }
}
void display()
{
        int i;
        for(i=0;i<cnt;i++)
        printf("%s\t%d\n",sy[i].label,sy[i].addr);
}
```

output:

// Example Usage

1. Insert Operation:

   Enter the label: A
   Enter the address: 100

   Enter the label: B
   Enter the address: 200

2. Display Operation:

   A 100
   B 200

3. Search Operation:

   Enter the label to search: A
   Label is found.

7. Eliminate Left Factoring .c

```
#include<stdio.h>
#include<string.h>
  int main()
  {
     char gram[20],part1[20],part2[20],modifiedGram[20],newGram[20],tempGram[20];
     int i,j=0,k=0,l=0,pos;
     printf("Enter Production : S->");
     gets(gram);
     for(i=0;gram[i]!='|';i++,j++)
         part1[j]=gram[i];
```

```
        part1[j]='\0';
        for(j=++i,i=0;gram[j]!='\0';j++,i++)
            part2[i]=gram[j];
        part2[i]='\0';
        for(i=0;i<strlen(part1)||i<strlen(part2);i++)
        {
            if(part1[i]==part2[i])
            {
                modifiedGram[k]=part1[i];
                k++;
                pos=i+1;
            }
        }
        for(i=pos,j=0;part1[i]!='\0';i++,j++){
            newGram[j]=part1[i];
        }
        newGram[j++]='|';
        for(i=pos;part2[i]!='\0';i++,j++){
            newGram[j]=part2[i];
        }
        modifiedGram[k]='X';
        modifiedGram[++k]='\0';
        newGram[j]='\0';
        printf("\n S->%s",modifiedGram);
        printf("\n X->%s\n",newGram);
    }
output:
Enter Production : S->ab|ac
S->aX
 X->b|c

6. Eliminate Left Recursion .c
#include<stdio.h>
#include<string.h>
#define SIZE 10
  int main () {
        char non_terminal;
        char beta,alpha;
        int num,i;
        char production[10][SIZE];
        int index=3; /* starting of the string following "->" */
        printf("Enter Number of Production : ");
        scanf("%d",&num);
        printf("Enter the grammar as E->E-A :\n");
```

```
    for(i=0;i<num;i++){
        scanf("%s",production[i]);
    }
    for(i=0;i<num;i++){
        printf("\nGRAMMAR : : : %s",production[i]);
        non_terminal=production[i][0];
        if(non_terminal==production[i][index]) {
            alpha=production[i][index+1];
            printf(" is left recursive.\n");
            while(production[i][index]!=0 && production[i][index]!='|')
                index++;
            if(production[i][index]!=0) {
                beta=production[i][index+1];
                printf("Grammar without left recursion:\n");
                printf("%c->%c%c\'",non_terminal,beta,non_terminal);
                printf("\n%c\'->%c%c\'|E\n",non_terminal,alpha,non_terminal);
            }
            else
                printf(" can't be reduced\n");
        }
        else
            printf(" is not left recursive.\n");
        index=3;
    }
 }
```

output:

Enter Number of Production : A -> a | aA | b

Enter the grammar as E->E-A :

5.Identifier or Not .C

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

int isValidIdentifier(const char *identifier) {
    if (!isalpha(identifier[0]) && identifier[0] != '_') {
        return 0; // Invalid: Identifier must start with a letter or underscore
    }

    for (int i = 1; i < strlen(identifier); i++) {
        if (!isalnum(identifier[i]) && identifier[i] != '_') {
            return 0; // Invalid: Contains non-alphanumeric characters other than underscore
        }
    }
```

```
    return 1; // Valid identifier
}

int main() {
    char identifier[30];

    printf("Enter an identifier: ");
    scanf("%s", identifier);

    if (isValidIdentifier(identifier)) {
        printf("Valid identifier\n");
    } else {
        printf("Invalid identifier\n");
    }

    return 0;
}
```

output:
Enter an identifier: myVariable123
Valid identifier

4.Number of whitespaces and newline characters.c

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

int main() {
    char c;
    int whitespaceCount = 0, newlineCount = 0;

    printf("Enter a string (Ctrl+D to end input):\n");

    while ((c = getchar()) != EOF) {
        if (isspace(c)) {
            if (c == '\n') {
                newlineCount++;
            } else {
                whitespaceCount++;
            }
        }
    }

    printf("\nNumber of whitespaces: %d\n", whitespaceCount);
```

```
    printf("Number of newline characters: %d\n", newlineCount);

    return 0;
}
```

output:

```
Enter a string (Ctrl+Z to end input):
Hello World!
This is a sample program.
Number of whitespaces: 5
Number of newline characters: 3
```

3.recognize the operators +,-,*,

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

int main() {
    char input[100];
    printf("Enter an arithmetic expression: ");
    fgets(input, sizeof(input), stdin);
// Tokenize the input string
    printf("Operators found: ");
    for (int i = 0; input[i] != '\0'; i++) {
        if (input[i] == '+' || input[i] == '-' || input[i] == '*' || input[i] == '/') {
            printf("%c ", input[i]);
        }
    }

    return 0;
}
```

output:

```
Enter an arithmetic expression: a+b*c
Operators found: + *
```

2 Comment or not. C

```
//CHECKING COMMENT OR NOT
#include<stdio.h>
#include<conio.h>
int main()
{
        char com[30];
        int i=2,a=0;
        printf("\nEnter Comment:");
        gets(com);
```

```
        if(com[0]=='/')
        {
                if(com[1]=='/')
                        printf("It is a comment\n");
        else if(com[1]=='*')
        {
                for(i=2;i<=30;i++)
                {
                        if(com[i]=='*'&&com[i+1]=='/')
                        {
                                printf("It is a comment\n");
                                a=1;
                                break;
                        }
                        else
                        continue;
                }
                if(a==0)
                        printf("\nIt is not a comment");
                }
                else
                        printf("\nIt is not a comment");
        }
        else
                printf("\n It is not a comment");
}
output:
Enter Comment:// This is a single-line comment
It is a comment

……recursive descent parsing
#include<stdio.h>
#include<conio.h>
#include<string.h>
char input[100];
int i,l;
int main()
{
printf("\nRecursive descent parsing for the following grammar\n");
printf("\nE->TE'\nE'->+TE'/@\nT->FT'\nT'->*FT'/@\nF->(E)/ID\n");
printf("\nEnter the string to be checked:");
gets(input);
if(E())
{
```

```
if(input[i+1]=='\0')
printf("\nString is accepted");
else
printf("\nString is not accepted");
}
else
printf("\nString not accepted");
getch();
}
E()
{
if(T())
{
if(EP())
return(1);
else
return(0);
}
else
return(0);
}
EP()
{
if(input[i]=='+')
{
i++;
if(T())
{
if(EP())
return(1);
else
return(0);
}
else
return(0);
}
else
return(1);
}
T()
{
if(F())
{
if(TP())
```

```
return(1);
else
return(0);
}
else
return(0);
}
TP()
{
if(input[i]=='*')
{
i++;
if(F())
{
if(TP())
return(1);
else
return(0);
}
else
return(0);
}
else
return(1);
}
F()
{
if(input[i]=='(')
{
i++;
if(E())
{
if(input[i]==')')
{
i++;
return(1);
}
else
return(0);
}
else
return(0);
}
else if(input[i]>='a'&&input[i]<='z'||input[i]>='A'&&input[i]<='Z')
```

```
{
i++;
return(1);
}
else
return(0);
}
```