Experiment: 1

P. Vishnu Vardhan Reddy
192011179

Aim: Implement a program to perform symbol table operations.

Program:

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <string.h>
#include <stdlib.h>
#define NULL 0
int size =0;
void Insert();
void Display();
void Delete();
Int search(char lab[]);
void modify();
struct symbTab
{
char label[10], symbol[10];
int addr;
struct symbTab *next;)
struct symbolTab *first, *last;
Intmain()
  {
```

P. Vishnu vardhan Reddy
19201179

```
int addr;
struct symbTab* next;};
int addr;
struct symbTab* first,* last.
int main()
{
int op,y; char lab[10];
//clrscr();
 do
{
printf("\n\t symbol table Implementation\n")
printf("\n\t1. Insert\n\t2. Display\n\t3. Delete\n\t4.
search\n\t5. modify\n\t6. END\n");
scanf("%d",&OP);
 switch(OP)
 {
case 1;
InsertA();
break.
case 2;
Display();
break
case 3;
Delete();
break;
case 4;
printf("\n\t Enter the label to be searched:")
scanf("%s",la);
```

P. Vithika.
192011179.

```
y=search(la)
printf("\n\t search result:");
if(y==1)
printf("\n\t the label is present in the symbol table\n")
else.
Printf("\n the label is not present in the symbol
        table \n")

break;
case 5
modify();
break;
case 6;
exit(0);
}
}
while(op<6);
getch();
}
void Insert()
{
int n;
char l[10];
printf("\n Enter the label!");
scanf("%s",l)
n=search(l);
(n==1)
printf("\n\t label exists already in the symbol table.\n\t
        Duplicate can't be Inserated")
else
}.
```

P. vithna. 192011179.

```
struct symbTab *p.
p= malloc(sizeof(struct symbTab));
strcpy(p->label,l);
printf("%s",p->symbol)
scanf("%d\n",Enter the address")
scanf("%d",&p->addr);
p-> next=null
if(size==0).
{
first scan.
printf("\n after modification:\n");
Display();
}
break;
case 2;
Printf("\n Enter the label where address is to be modified")
scanf("%s",I);
s=search(I)
if(s==0)
Printf("\n Enter the new address")
scanf("%d",&address)
for(i=0; i<size; i++)
{
if(strcmp(p->label,I)==0.
p-> addr =add.
P=P->next.
```

P. Vishnu.
192011479.

```
{
int a;
char l[10];
struct symbTab *p,*q;
p=first.
printf("\n Enter the label to be deleted:");
scanf("%s",l);
a=search(l); if(a==0)
printf("\n label not found\n")
else.
{
if(strcmp(first->label,l)==0.
 first=first->next.
else if(strcmp(q->label,l)!=0)
{
p=p->next.
```

Sample output:-

Symbol Table Implemenation.

Insert
Display.
search.
modify.
END.

Enter your option 1:
Enter your label Plus.
Enter the symbol: t
Enter the addres: 100
Label Inserted.

symbol table Implemenation.

Inserct.
Display
Delete.
modify
END.

Enter the option :2
label symbol . Address
Plus t 100.

Symbol Table Implemenation.

Insert.
Display
Delete.
Search.
modify
END.

Enter you roption : 5.

what do you want to option ! only the label.
only the address.
Both the label and address Enter your choice.
Enter the old label : minus.
Enter the new label : minus : new.
After modification.

label	symbol	address.
minus-new	—	200

symbol table Implemonation.

Inserel.
Display
Delete.
Search.
modify. Enter you roption : 6.
END.

Result; Hence the Implemenation, c program to perform symbol table operation.

Name
Regno 111179.

2: Experiment: 2

P. Vishnu
(9201179.

Aim:- To write c program for identify Identifier constants and operations.

Program:

```
#include<string.h>.
#include<type.h>.
#include<stdio.h>.
void keyword (char str[100];
{
if (strcmp ("for",str) ==0||strcmp("while",str) ==0||
strcmp("do",str) ==0 strcmp("int",str)==0||strcmp.
(float",str) ==0||strcmp("char",str)||strcmp("double.
",str) ||strcmp("str printf("\n is a key board");
printf("\n%s is an identifier",str);
}
main();
{
file *f1*f2*f3;
char c,str[10],str[12];
int num[100] line no=0; token=0, i=i==0;k=0;
printf("\n Enter the c program:")
(*gets (str);
```

```
fs=fopen("special char","w")
while((c=getc(fi))!=EOF)
{
if=fopen("input","r");
fs=fopen(c=getc(fi))!=EOF)
{
if(isdigit(c))
{
token value=c-'0';c=getc(fi);
while(isdigit(c));
{.
token value*=10+c-'0';
c=getc(fi);
}
num(i++)=token value;
ungetc(fi);
}
num(i++)=token value;
ungets(c,f);
}
else if(isalph(c));
{
Put(i,fs);c=getc(fi);
while(isdigit(c)alpha(c)!=(c-5);
```

S. Vishnu.
192011179.

P. Vishnu.
1920111fa.

```
}
puts("f2")
ungetc(c,f1);
}
else
if{c=='('||c==')'||c=='{'||c=='}')
printf("\n")
else if(c=='\n') lineno++;
else
putc(c,f3);
}
fclose(f2);
fclose(f3);
fclose(f1);
printf("\n\n The no's in the program are\n")
for(i=0; i<k; i++);
printf("\n\n")
f2=fopen("identifier","r").
  k=0.
printf("\n\n The keywords and identifiers are\n").
while(c=getc(f2))!=EOF).
  {
if(c!=getc(f2)!=EOF).
}
else
str(k)="\0"
keywords[str];
 k=0;
```

```
}
}
fclose(f2);
f3=open("special char","r");
printf("\n special characters are");
while(c=getf f3!=EOF).
printf("\n")
fclose(f3)
printf("\n total no. of lines are %d",lines);
}.
```

output:

Input.

Enter program (ctrl D for termination);

```
{
int a[3],t1,t2.
t1=2; a[0]=1; a[1]=2; a[t]=3;
t2=(a(2)+t1%6)(a(-2)-t);
if t2>5 then printf(t2); else int t3; t3=99, t2=22;
printf t+t1+t+t2+t3; /*this is comment on 2 lines.
}
```

}. output:-

Variables:- a[3] t2 t3 operator:- +*/ constant.

keywords:- int if then else end if special symbol

comment:- this is comment on 2 lines.

Result:-

3.

Experiment : 3

P. Vishnu
Regno:- 192011479

Aim:- Implement a C program to elimate left-recursion from to given CFG.

Program:-

```
#include<stdio.h>
#include<string.h>
#define size 10
int main(){
char beta,alpha;
int num;
char production[10][size];
index=3;
/* string of the string following "->"*/
printf("Enter the number of production:");
scanf("%d",&num);
printf("Enter the grammer as E->E-A:/n");
for(int i=0;i<num;i++)
{
scanf("%s",production[i]);
}
for(int=i=0;i<num;i++);
printf("\n Grammer::::%s",production[i]);
non terminal=production[i][index];
alpha=production[i][index+1];
printf("\n is left recursive");
```

```
}
for(int i=0; i<n; i++);
{
printf("\n Grammer ::%s production(%d)",
non terminal=production[i][index];
alpha=production[i][index+1];
printf("\n is a left recursive (n");
non terminal=production[i][index];
alpha=production[i][index+1];
printf("\n is left recursive (n");
while(production[i][index]!='\0' && production[i]
index!=Grammer without left recursion.
else.
printf("\n Cant be reduction(n");
}
else.
printf("\n is not left recursive(n");indexes
}
}
```

sample output:-

Enter the number of production :- 4 Enter the grammer as E->EA; E->E/A
A->AT/a T->a.
E->i

Grammer :- E->EA/A is left recursive Grammer without
E->AE'/E->AE'/E.

Grammer :- A->AT/a is left recursive without left recursion.

14. Left factoring Python. Reg: 19201179.

Aim: To Implement a C program to Eliminate left factoring from a given CFG.

Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
char gram[20], part[20], part[20], modified[20];
 int i, j=0, k=0, j=0, pos;
printf("Enter production: A->");
gets(gram);
part[0]=gram[0];
for(i=0; i<strlen(gram); i++,j++)
{
if part[j]==part2[j]
{
modified gram[k]=part[j++];
 pos=i+1;
}
}
new grand(j++)='|';
for(i=pos; i=0; part[i]; i++)
```

output: Enter production A->abcn=|abcFG.

A->AbcX X->DE|FG.

Result: C program for left factoring is Executed.