7. Write a C program to recognize strings under ‘a*’, ‘a*b+’, ‘abb’.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
int main()
{
char ch[100];
int i,state;
printf("ENTER THE STRING\t");
gets(ch);
i=0;
state=0;
while(ch[i]!='\0')
{
switch(state)
{
case 0:if(ch[i]==' ') state=0;
else if(ch[i]=='a')
state=1;
else if(ch[i]=='b')
state=2;
else
state=6;
break;
case 1:if(ch[i]=='a')
state=3;
else if(ch[i]=='b')
state=4;
else
state=6;
break;
```

```
case 2:if(ch[i]=='a')
state=6;
else if(ch[i]=='b')
state=2;
else
state=6;
break;
case 3:if(ch[i]=='a')
state=3;
else if(ch[i]=='b')
state=2;
else
state=6;
break;
case 4:if(ch[i]=='a')
state=6;
else if(ch[i]=='b')
state=5;
else
state=6;
break;
case 5:if(ch[i]=='a')
state=6;
else if(ch[i]=='b')
state=2;
else
state=6;
break;
case 6:printf("NOT RECOGNISED\n");
return(main());
}
i++;
```

```
}
if(state==0 || state ==1 || state==3)
printf("BELONG TO PATTERN a*\n");
else if(state==2 || state==4)
printf("BELONG TO PATTERN a*b+\n");
else if(state==5)
printf("BELONG TO PATTERN abb\n");
else
printf("NOT RECOGNISED\n");
return (main());
}
```

8. Write a C program to test whether a given identifier is valid or not.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
int main()
{
int i;
 char id[100];
 printf("enter the string\n");
 gets(id);
 if(id[0]!='_'&&!isalpha(id[0]))
 {
 printf("it is not a valid c identifier1\n");
 return 0;
 }
 for(i=1;id[i]!='\0';i++)
 {
 if(isalpha(id[i])||isdigit(id[i])||id[i]=='_')
 {
 continue;
```

```
}
else
{
printf("It is not a valid c identifier\n");
return 0;
}
}
printf("It is a valid c identifier\n");
return 0;
}
```

9. Write a C program to compute FIRST of all Non Terminals of a given grammar.

```
#include<stdio.h>
#include<ctype.h>
void FIRST(char[],char );
void addToResultSet(char[],char);
int numOfProductions;
char productionSet[10][10]; char C; int recur;
main()
{    int i;
    char choice,c,result[20];
    printf("How many number of productions:");
    scanf(" %d",&numOfProductions);
    printf("\n Enter production like this eg: E=E+T   Enter # for
epsilon \n");
    for(i=0;i<numOfProductions;i++)
    {printf("Enter productions Number %d : ",i+1);
     scanf(" %s",productionSet[i]);
    }
    do
    {printf("\n Find the FIRST of  :");
     scanf(" %c",&c);C=c;recur=0;
     FIRST(result,c);
     //Compute FIRST; Get Answer in 'result' array
     printf("\n FIRST(%c)= { ",c);
     for(i=0;result[i]!='\0';i++)
     printf(" %c ",result[i]);       //Display result
     printf("}\n");
      printf("press 'y' to continue : ");
     scanf(" %c",&choice);
    }    while(choice=='y'||choice =='Y');}

void FIRST(char* Result,char c)
{    int i,j,k,foundEpsilon; char subResult[20];
    subResult[0]='\0'; Result[0]='\0';
    //If X is terminal, FIRST(X) = {X}.
    if(islower(c)||!(isalpha(c)))
```

```
    {  addToResultSet(Result,c); return ; }

    for(i=0;i<numOfProductions;i++)
    {
      if(productionSet[i][0]==c)
      {
      if(productionSet[i][2]=='#')
               addToResultSet(Result,'#');
       else
           { j=2;
            while(productionSet[i][j]!='\0')
            {foundEpsilon=0;recur++;
            FIRST(subResult,productionSet[i][j]);
            for(k=0;subResult[k]!='\0';k++)
            if(!(recur>1 && subResult[k]=='#'&&
productionSet[i][j+1]!='\0'))
                addToResultSet(Result,subResult[k]);
                for(k=0;subResult[k]!='\0';k++)
                if(subResult[k]=='#')
                {   foundEpsilon=1; break;   }

                if(!foundEpsilon) break;
                j++;
                }  } } }    return ; }

void addToResultSet(char Result[],char val)
{    int k;
    for(k=0 ;Result[k]!='\0';k++)
        if(Result[k]==val) return;
    Result[k]=val; Result[k+1]='\0';
}
```

11. Write a C program to implement recursive descent parsing for the given grammar.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
void Nonterminal(char );
int numOfProductions,k,temp;
char productionSet[10][10]; char str[20]; int reult;
main()
{
    int i,ch;
    printf("How many number of productions:");
    scanf(" %d",&numOfProductions);
    printf("\n Enter production like this eg: E->E+T   Enter #  for
epsilon \n");
    for(i=0;i<numOfProductions;i++)
    {
      printf("Enter productions Number %d : ",i+1);
      scanf(" %s",productionSet[i]);
    }
       do
       {
```

```
      k=0;
      printf("\n Enter the string \n");
      scanf(" %s",&str);
      Nonterminal(productionSet[0][0]);
      if(k==strlen(str))
      printf("\ninput string is valid");
      else
      printf("\ninput string is not valid");
      printf("\nDo you want to continueor not 1/2");
      scanf("%d",&ch);
      } while(ch==1);
      return 0;
}

void Nonterminal(char p)
{
    int i,j,found=0;


    for(i=0;i<numOfProductions;i++)
    {
     temp=k;
     if(productionSet[i][0]==p)
     {
           for(j=3;productionSet[i][j]!='\0';j++)
           if(isupper(productionSet[i][j]))
           {
           found=1;
           Nonterminal(productionSet[i][j]);
           }
           else if(productionSet[i][j]==str[k])
           { k++;
           found=1;
           }
           else if(productionSet[i][j]=='#')
               {
               found=1;
               return;
               }
           else
           {
           k=temp;  break;
           }
       }}

      if((i>=numOfProductions) && (found==0) && (k!=strlen(str)))
       {
       printf("\ninput is not Valid");
       exit(0);
       }
      }
```

10. Write a C program to construct predictive parsing table for the given grammar.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
void addtononT(char);
void addtoter(char);
int nop,ppt[10][10];
char
productions[10][10],ter[10],nont[10],first[10][10],follow[10][10];
int main()
{
int i,j,k,m,pos=0;
for(i=0;i<10;i++)
for(j=0;j<10;j++)
ppt[i][j]=-1;
printf("Enter the number of productions:");
scanf("%d",&nop);
printf("\nEnter production like this eg:E->E+T Enter # for
epsilon\n");
for(i=0;i<nop;i++)
{
printf("Enter production number %d:",i+1);
scanf("%s",productions[i]);
}
for(i=0;i<nop;i++)
addtononT(productions[i][0]);
for(i=0;i<nop;i++)
for(j=3;productions[i][j]!='\0';j++)
if(islower(productions[i][j])||(!isalpha(productions[i][j])))
addtoter(productions[i][j]);
for(j=0;ter[j]!='\0';j++);
ter[j]='$';
ter[++j]='\0';
printf("Enter first of all non terminals without any space b/w the
symbols like abc#,#for epsilon\n" );
for(i=0;i<nop;i++)
{
printf("Enter first of:");
for(k=3;k<productions[i][k]!='\0';k++)
printf("%c",productions[i][k]);
printf("=");
scanf("%s",first[i]);
for(j=strlen(first[i]);j>=0;j--)
first[i][j+1]=first[i][j];
first[i][0]=productions[i][0];
}
printf("Enter follow of all non terminals without any space b/w
symbols like abc#, # for epsilon\n");
for(i=0;nont[i]!='\0';i++)
{
printf("Enter follow of %c=",nont[i]);
scanf("%s",follow[i]);
for(j=strlen(follow[i]);j>=0;j--)
```

```
follow[i][j+1]=follow[i][j];
follow[i][0]=nont[i];
}
for(i=0;i<nop;i++)
{
for(m=0;follow[m][0]!=first[i][0];m++);
for(j=1;first[i][j]!='\0';j++)
if(first[i][j]!='#')
{
for(k=0;ter[k]!='\0';k++)
if(ter[k]==first[i][j])
break;
ppt[m][k]=i;
}
else
{
for(m=0;follow[m][0]!=first[i][0];m++);
for(j=1;follow[m][j]!='\0';j++)
{
for(k=0;ter[k]!='\0';k++)
if(ter[k]==follow[m][j])
break;
ppt[m][k]=i;
}
}
first[i][0]='0';
}
printf("Predictive parsing table\n");
printf("..................Terminals.................\n");
printf("Non Terminals |\t\t");
for(i=0;ter[i]!='\0';i++)
printf("%c\t",ter[i]);
printf("\n");
for(i=0;follow[i][0]!='\0';i++)
{
m=0;
printf("%c\t\t",nont[i]);
for(j=0;ter[j]!='\0';j++)
{
pos=ppt[i][j];
for(;m<=j;m++)
printf("\t");
if(pos!=-1)
printf("%s",productions[pos]);
}
printf("\n");
}
return 0;
}
void addtononT(char c)
{
int j;
for(j=0;nont[j]!='\0';j++)
if(nont[j]==c)
return ;
nont[j]=c;
```

```
nont[j+1]='\0';
}
void addtoter(char c)
{
int j;
for(j=0;ter[j]!='\0';j++)
if(ter[j]==c)
return ;
if(c!='#')
{
ter[j]=c;
ter[j+1]='\0'; }}
```

12. Program to construct the closure of an LR(0) item with respect to the given grammar

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void closure(char []);
char citem[10][10],gram[10][10];int nop,noi;
int main()
{
int i,ch;
char agram[10][10],item[10];
printf("How many number of productions:");
scanf(" %d",&nop);
printf("\n Enter production like this eg: E->E+T Enter #  for
epsilon \n");
for(i=0;i<nop;i++)
{
printf("Enter productions Number %d : ",i+1);
scanf("%s",&gram[i]);
}
agram[0][0]=gram[0][0];
agram[0][1]='!';
agram[0][2]='-';
agram[0][3]='>';
agram[0][4]=gram[0][0];
agram[0][5]='\0';
for(i=1;i<=nop;i++)
strcpy(agram[i],gram[i-1]);
printf("\n Augmented grammar is : ");
for(i=0;i<=nop;i++)
printf(" %s ",agram[i]);
do
{
printf("\n Enter the item to find the closure \n");
scanf("%s",&item);
printf("Closure of  %s = {",item);
closure(item);
for(i=0;i<noi;i++)
printf("%s    ",citem[i]);
```

```
printf("}\n");
printf("\n Enter 1:To Continue 2: To Stop \n");
scanf("%d",&ch);
}
while(ch==1);
return 0;
}

void closure(char it[20])
{
int i,j,l,k=0,found;char temp[10];
/* Rule 1: Add everythinmg in I to closure of I */
noi=0;
strcpy(citem[k],it);
noi++;
while(k<noi)
{
i=0;
while(it[i]!='\0' && it[i]!='.')/*To traverse upto . in ITEM */
i++;
if(i<(strlen(it)-1))
{
for(j=0;j<nop;j++)
{
found=0;
if(it[i+1]==gram[j][0] && isupper(it[i+1]))
{
strcpy(temp,gram[j]);
for(l=strlen(temp);l>=3;l--)
temp[l+1]=temp[l];
temp[l+1]='.';

for(l=0;l<noi;l++)
if(strcmp(citem[l],temp)==0)
found=1;
if(found==0)
{
strcpy(citem[l],temp);
noi++;
}
}
}
}
else
{
printf(" %s }",it);
exit(0);
}
k++;
strcpy(it,citem[k]);
}
return;
}
```