Lab 5: CFG

1) Implement the following CFG, in the way shown above or with the help of a user defined stack, that is, in the way it may work in an implementation of the transition function of a PDA.

A → aXd

X → bbX

X → bcX

X → □

```
//CFG

#include <iostream>
#include <string>
#include <cctype>
#include <algorithm>

using namespace std;

string str;
bool found;
int length, i;

void A();
void X();

// X -> bbX | bcX | E
void X() {
    if(i<length && str[i]=='b'){
        i++;
        if(i<length && (str[i]=='b' || str[i]=='c')){
            i++;
            X();
            return;
        }else{
            found=false;
            return;
        }
    }
    found=true;
}

// A -> aXd
void A() {
    if(i<length && str[i]=='a'){
        i++;
        X();
        if(!found)  return;

        if(i<length && str[i]=='d'){
```

```
            i++;
            found=true;
        }else{
            found=false;
        }
    }else{
        found=false;
    }
}

int main() {
    while(1){
        cout << "Enter a string(type 'exit' to quit):\n";
        cin>>str;
        if(str=="exit") break;

        i=0;
        length=str.size();
        found=true;

        A();

        if(found && i == length){
            cout << "Accepted" << endl;
        } else {
            cout << "Rejected" << endl;
        }
    }
    return 0;
}
```

---

2)

```
#include<iostream>
#include<algorithm>
#include<string>
#include<cctype>

using namespace std;

string str;
bool found;
int length,i;

void A();
void B();
void C();

// C -> eeC | E
void C()
{
    if(i<length && str[i]=='e'){
        i++;
        if(i<length && str[i]=='e'){
            i++;
            C();
            return;
```

```
        }else{
            found=false;
            return;
        }
    }
     found=true;
}

//B -> cB | dC
void B()
{
    if(i<length && str[i]=='c'){
        i++;
        B();
        return;
    }else if(i<length && str[i]=='d'){
        i++;
        C();
        if(!found)  return;
    }else{
        found=false;
    }
}

//A ->  bB
void A()
{
    if(i<length && str[i]=='b'){
        i++;
        B();
        if(!found)  return;
    }else{
        found=false;
    }
}

int main()
{
    while(1){
        cout<<"Enter a string(Type 'exit' to quit): "<<endl;
        cin>>str;
        if(str=="exit") break;

        i=0;
        length=str.size();
        found=true;

        A();

        if(found && i==length){
            cout<<"Accepted"<<endl;
        }else{
            cout<<"Rejected"<<endl;
        }
    }
}
```

Lab 1

3) Extracting the sequence of occurrences of a specified character from a source program.

```
#include<bits/stdc++.h>
#include<math.h>
using namespace std;

int main(void)
{
    FILE *p1, *p2;
    char c;

    p1=fopen("datafile1.cpp","r");
    p2=fopen("parentheses.txt","w");

    if(!p1)
        cout<<"\nFile can't be opened!"<<endl;
    else{
        while((c=fgetc(p1)) != EOF){
            if((c=='(') || (c==')'))
                fputc (c, p2);
        }
    }

    fclose(p1);
    fclose(p2);

    p2=fopen("parentheses.txt","r");
    while((c=fgetc(p2)) != EOF)
        cout<<c;

    fclose(p2);

    return 0;
}
```

4) Write a program to print the header files used in a source program.

```
#include<bits/stdc++.h>
using namespace std;

int main(void)
{
    FILE *p1;
    char c;

    p1=fopen("datafile1.cpp","r");

    if(!p1)
        cout<<"\nFile can't be opened!"<<endl;
    else{
            string line="";
            while((c=fgetc(p1))!=EOF){
                if(c!='\n'){
```

```
                    line+=c;
                }
                else{
                    if(line.find("#include")==0){
                        size_t start=line.find('<');
                        size_t end=line.find('>');
                        if(start!=string::npos && end!=string::npos &&
end>start){
                            cout<<line.substr(start+1,end-start-1)<<endl;
                        }
                    }
                    line="";
                }
            }
            if(!line.empty() && line.find("#include") == 0){
                size_t start = line.find('<');
                size_t end = line.find('>');
                if (start != string::npos && end != string::npos && end >
start) {
                    cout << line.substr(start + 1, end - start - 1) << endl;
                }
            }
    }

    fclose(p1);
    return 0;
}
```

5) Write a program to add line numbers to a source program.

```
#include<bits/stdc++.h>
using namespace std;

int main(void)
{
    FILE *p1;
    char line[1000];
    int lineNumber=1;

    p1=fopen("datafile1.cpp","r");

    if(!p1)
        cout<<"\nFile can't be opened!"<<endl;
    else{
            while(fgets(line,sizeof(line),p1)){
                cout<<lineNumber<<": "<<line;
                lineNumber++;
            }
    }

    fclose(p1);
    return 0;
}
```

Lab 2

6) Write a program to recognize whether the entered string is a keyword or not.

```
#include<bits/stdc++.h>
using namespace std;

bool isKeywordDFA(string str)
{
    int state=0;
    int i=0;

    while(i<str.length()){
        char ch=str[i];

        switch(state){
            case 0:
                if(ch=='i')      //if, int
                    state=1;
                else if(ch=='e')     //else
                    state=10;
                else if(ch=='r')     //return
                    state=20;
                else
                    return false;
                break;

            case 1:
                if(ch=='f')
                    state=2;
                else if(ch=='n')
                    state=3;
                else
                    return false;
                break;

            case 2:
                return (i==str.length()-1);

            case 3:
                if(ch=='t')
                    state=4;
                else
                    return false;
                break;

            case 4:
                return (i==str.length()-1);

            case 10:
                if(ch=='l')
                    state=11;
                else
                    return false;
```

```
        break;

    case 11:
        if(ch=='s')
            state=12;
        else
            return false;
        break;

    case 12:
        if(ch=='e')
            state=13;
        else
            return false;
        break;

    case 13:
        return (i==str.length()-1);

    case 20:
        if (ch == 'e')
            state = 21;
        else
            return false;
        break;

        case 21:
        if (ch == 't')
            state = 22;
        else
            return false;
        break;

    case 22:
        if (ch == 'u')
            state = 23;
        else
            return false;
        break;

    case 23:
        if (ch == 'r')
            state = 24;
        else
            return false;
        break;

    case 24:
        if (ch == 'n')
            state = 25;
        else
            return false;
        break;

    case 25:
        return (i == str.length() - 1);
```

```
                default:
                    return false;
            }
            i++;
        }
        return (state == 2 || state == 4 || state == 13 || state == 25);
    }

    int main()
    {
        string input;
        cout<<"Enter a string: ";
        cin>>input;

        if(isKeywordDFA(input))
            cout << "\"" << input << "\" is a valid keyword." << endl;
        else
            cout << "\"" << input << "\" is NOT a keyword." << endl;

        return 0;
    }
```

7) Write a program to detect whether the entered string is an identifier or not based on the finite automata.

```
#include<bits/stdc++.h>
using namespace std;

bool isIdentifier(string str)
{
    int state=0;

    if(str.empty())
        return false;

    for(int i=0;i<str.length();i++)
    {
        char ch=str[i];

        switch(state)
        {
            case 0:
                if(isalpha(ch) || ch=='_')
                    state=1;
                else
                    return false;
                break;

                case 1:
                if(isalnum(ch) || ch=='_')
                    state=1;
                else
                    return false;
                break;
```

```
        }
    }
    return (state==1);
}

int main()
{
    string input;
    cout<<"Enter a string: ";
    cin>>input;

    if(isIdentifier(input))
        cout << "\"" << input << "\" is a valid identifier." << endl;
    else
        cout << "\"" << input << "\" is NOT a valid identifier." << endl;

    return 0;
}
```

8) Write a program to detect whether the entered string is a valid Number or not.

```
#include<bits/stdc++.h>
using namespace std;

bool isNumber(string str)
{
    int state = 0;

    for(int i=0; i<str.length(); i++){
        char c = str[i];

        switch(state)
        {
            case 0: // start
                if(isdigit(c))
                    state = 1;   // integer part
                else
                    return false;
                break;

            case 1: // integer part
                if(isdigit(c))
                    state = 1;   // stay in integer part
                else if(c=='.')
                    state = 2;   // decimal point
                else
                    return false;
                break;

            case 2: // after decimal point
                if(isdigit(c))
                    state = 3;   // fractional part
                else
                    return false;
                break;
```

```
            case 3: // fractional part
                if(isdigit(c))
                    state = 3;   // stay in fractional part
                else
                    return false;
                break;
        }
    }

    // Valid end states = integer part (1) or fractional part (3)
    return (state==1 || state==3);
}

int main()
{
    string input;
    cout<<"Enter an input: ";
    cin>>input;

    if(isNumber(input))
        cout << "\"" << input << "\" is a valid Number" << endl;
    else
        cout << "\"" << input << "\" is NOT a valid Number" << endl;

    return 0;
}
```