```
#include<iostream>
#include<cstring>
                std;
# define Maxsize 10000

      stack
{
      :
    int top;
    string arr[Maxsize];
    int IsEmpty()
    {
           (top <0)
                   1;

                   0;
    }
    int IsFull()
    {
           (top==Maxsize-1)
                   1;

                   0;
    }
    void push(string v)
    {
           (IsFull()==1)
            cout<< "Stack is full"<< endl;

            {top=top+1;
            arr[top]=v;}
    }
    void pop()
    {
           (IsEmpty()==1)
            cout << "Stack is empty" << endl;

            top--;
    }
};

stack stack_c;

      all_string
{
      :

    string rule_matrix[Maxsize][Maxsize];
    string look_up[Maxsize][Maxsize];
    string input[Maxsize];
    string sym[Maxsize];
    int rules_applied[Maxsize];
};

all_string M;

int get_c(string a[], string s ){
    int i=0;
         (a[i]!= s)
        i=i+1;
           i;
}

int check(string a, string b){
       (a==b)
               1;

               0;
}

int power(int a,int b){
       (b==0)
```

```
               1;

               a*power(a,b-1);
}

int number(string a, int initial){
    int size=a.size()-1;
    int num=0;
    int pow=-1;
       (int i=size;i>initial;i--)
        {pow=pow+1;
        num=num+(a[i]-'0')*power(10,pow);}
           num;
}

string no_to_string(int number){
       (number == 0)
               "0";
    string temp="";
    string returnvalue="";
          (number>0)
    {
        temp+=number%10+48;
        number/=10;
    }
        (int i=0;i<temp.length();i++)
        returnvalue+=temp[temp.length()-i-1];
           returnvalue;
}

int size_arr_string(string a[]){
    int count=0;

       (int i=0;i<Maxsize;i++)
        {  (check(a[i]," ")==0 && check(a[i],"")==0)
                  count=count+1;}
           count;
}

int Rule(int i){
    int size=size_arr_string(M.rule_matrix[i-1])-1;
    int count;
        (int top=stack_c.top;top>=0;top--)
        {
        count = 0;
            (int j=0;j<size-1;j++)
            {
            count=count+check(M.rule_matrix[i-1][size-j],stack_c.arr[top-2*j-1]);}
           (count == (size-1))
            {stack_c.top=top- 2*(size-1);
                 ;}
        }
    stack_c.push(M.rule_matrix[i-1][0]);
    int r=number(stack_c.arr[stack_c.top-1],-1);
    int c=get_c(M.sym,M.rule_matrix[i-1][0]);
    int state=number(M.look_up[r][c],-1);
    stack_c.push(no_to_string(state));
           state;
}

int final_task(int n){
    int r,c,state;
    r=0;
    string to_do;
    int count_rule=0;
    int flag;
    stack_c.push("0");
        (int i=0;i<=Maxsize;i++)
        {   c=get_c(M.sym,M.input[i]);
            to_do=M.look_up[r][c][0];
               (to_do=="p")
                {state=number(M.look_up[r][c],0);
                stack_c.push(M.input[i]);
                stack_c.push(no_to_string(state));
                }
                    (to_do=="r")
                {state=Rule(number(M.look_up[r][c],0));
                i=i-1;
                int v=number(M.look_up[r][c],0);
                M.rules_applied[count_rule]=v;
                count_rule=count_rule+1;}
                    (to_do=="a")
                {cout << "accepted"<<endl;
                flag=1;
                     ;}

                {cout<<"rejected"<<endl;
                flag=0;
                     ;}
            r=state;
        }

    int rev_rule[Maxsize];
        (int i=0;i<count_rule;i++)
        {rev_rule[count_rule-i-1]=M.rules_applied[i];}

       (flag==1)
    {    (int i=0;i<count_rule;i++)
        {
           (int j=0;j<Maxsize;j++)
            cout<<M.rule_matrix[rev_rule[i]-1][j];
        cout<<endl;}
    }
}

main(){
    int no_rule;
    cin>> no_rule;
    cin.ignore();

    char arr[no_rule][Maxsize];

       (int i=0 ; i<no_rule ; i++)
        {
        cin.getline(arr[i],Maxsize);
        }
    int n_count;
       (int i=0 ; i<no_rule ; i++)
        {
        n_count =0;
        char* temp_char;
        temp_char=strtok(arr[i]," ");

              (temp_char != NULL)
            {
            M.rule_matrix[i][n_count]=temp_char;
            temp_char = strtok(NULL, " ");
            n_count=n_count+1;
            }
        }

    int f,n_f;
    cin>> f;
    cin.ignore();
    cin >> n_f;
    cin.ignore();
    int t_sym=f+n_f;
       (int i=0;i<t_sym;i++)
        {cin>> M.sym[i];
        cin.ignore();}
    int r,c;
    cin >>r;
    cin.ignore();
    cin >>c;
    cin.ignore();
       (int i=0;i<r;i++)
    {
           (int j=0;j<c;j++)
            {
            cin >> M.look_up[i][j];
            cin.ignore();
            }
    }
    int no_seq;
    cin>>no_seq;
    cin.ignore();
    stack_c.top=-1;
       (int i=0;i<no_seq;i++)
        {
        stack_c.top=-1;
        int n;
            (int i=0;i<Maxsize; i++)
            {cin>> M.input[i];
            cin.ignore();
               (M.input[i]=="$")
                {n=i;
                     ;}
            }
        final_task(n);
        }
}
```