Problem-1

```
#include<iostream>
#include<string>
using namespace std;
#define MAX_INT_SIZE 10000
struct SeqStack
{
    int *top;
    int *base;
    int stacksize;
};
SeqStack s;
bool IsEmpty(SeqStack &s)
{
    if (s.top == s.base)
        return true;
    else
        return false;
}
bool IsFull(SeqStack&s)
{
    if (s.top - s.base >= s.stacksize)
        return true;
    else
        return false;
}
void PopStack(SeqStack &s)
{
    if (IsEmpty(s))
        cout << "Stack is Empty" << endl;
    else
        cout << *--s.top << endl;
}
void PushStack(SeqStack &s, int& e)
{
    if (IsFull(s))
        cout << "Stack is Full" << endl;
    else
```

```
        *s.top++ = e;
}
void PrintStack(SeqStack &s)
{
    while (s.top != s.base)
        cout << *--s.top << " ";
}
int main()
{
    int n;
    cin >> n;
    s.base = new int;
    s.top = new int;
    s.top = s.base;
    s.stacksize = n;
    string s1;
    int x;
    int y[100] = { 0 };
    string vis[100];
    int i = 0;
    while (1)
    {
        cin >> s1;
        if (s1 == "pop")
            vis[i] = "pop";

        else if (s1 == "push")
        {
            cin >> x;
            vis[i] = "push";
            y[i] = x;
        }
        else
        {
            vis[i] = "quit";
            break;
        }
        i++;
    }
    int k = 0;
    while (vis[k] != "quit")
    {
        if (vis[k] == "pop")
            PopStack(s);
        else
            PushStack(s, y[k]);
        k++;
    }
    PrintStack(s);
    return 0;
}
```

Problem-2

```
#include<iostream>
#include<cstring>
using namespace std;
#define MAX_INT_SIZE 10000
struct SeqStack
{
    char *top;
    char *base;
};
SeqStack s;
void PopStack(SeqStack &s)
{
    cout << *--s.top << endl;
}
void PushStack(SeqStack &s, char e)
{
    *s.top++ = e;
}
bool IsEmpty(SeqStack &s)
{
    if (s.top == s.base)
        return true;
    else
        return false;
}
void PrintStack(SeqStack &s)
{
    while (s.top != s.base)
```

```
        cout << *--s.top;
}
void convert_mTOn(SeqStack &s, int m, int n, char str[])
{
    int x, temp = 0, y = 1;
    for (int i = 0; i < strlen(str); i++)
        PushStack(s, str[i]);
    while (!IsEmpty(s))
    {
        s.top--;
        if (*s.top >= '0'&&*s.top <= '9')
            x = *s.top - '0';
        if (*s.top >= 'A'&&*s.top <= 'Z')
            x = *s.top - 'A' + 10;
        temp = temp + y * x;
        y *= m;
    }
    while (temp)
    {
        int r = temp % n;
        PushStack(s, (r <= 9 ? '0' + r : 'A' + r - 10));
        temp /= n;
    }
}
int main()
{
    int m, n;
    char str[100];
    s.top = new char;
    s.base = new char;
    s.top = s.base;
    cin >> m >> n;
    cin >> str;
    convert_mTOn(s, m, n, str);
    PrintStack(s);
    cout << endl;
    return 0;
}
```

Problem-3

```
#include<iostream>
#include<cstring>
using namespace std;
#define MAX_INT_SIZE 10000
struct SeqStack
{
    char *top;
    char *base;
};
SeqStack s;
void PopStack(SeqStack &s)
{
    s.top--;
}
void PushStack(SeqStack &s, char e)
{
    *s.top++ = e;
}
bool IsEmpty(SeqStack &s)
{
    if (s.top == s.base)
        return true;
    else
        return false;
}
void PrintStack(SeqStack &s)
{
    while (s.top != s.base)
        cout << *--s.top;
}
void Match(SeqStack &s, char s0[])
{
    int i = 0;
    if (s0[0] == '}' || s0[0] == ']' || s0[0] == ')')
    {
        cout << "no" << endl;
        cout << s0[0] << "期待左括号" << endl;
        return;
    }
    else
    {
        PushStack(s, s0[0]);
        i++;
        while (s0[i] != '\0')
        {
            int flag = 0;
            if ((s0[i] == '}'&&*(s.top - 1) == '{') || (s0[i] ==
']'&&*(s.top - 1) == '[') || (s0[i] == ')'&&*(s.top - 1) == '('))
            {
                PopStack(s);
                flag = 1;
            }
            else if (s0[i] == '{' || s0[i] == '[' || s0[i] == '(')
                PushStack(s, s0[i]);
            else if (((s0[i] == '}'&&*(s.top - 1) != '{') || (s0[i] ==
']'&&*(s.top - 1) != '[') || (s0[i] == ')'&&*(s.top - 1) != '('))
&& !IsEmpty(s))
            {
                cout << "no" << endl;
                cout << *--s.top << "期待右括号" << endl;
                return;
            }
            else if (IsEmpty(s) && (s0[i + 1] != '\0') && (s0[i] == '}'
|| s0[i] == ']' || s0[i] == ')'))
            {
                cout << "no" << endl;
                cout << s0[i] << "期待左括号" << endl;
                return;
            }
            i++;
            if (flag == 1 && s0[i] != '\0')
            {
                if (s0[i] == '{' || s0[i] == '[' || s0[i] == '(')
                {
                    PushStack(s, s0[i]);
                    i++;
                }
                if (IsEmpty(s) && s0[i + 1] == '\0' && (s0[i] == ')' ||
s0[i] == '}' || s0[i] == ']'))
```

```
                {
                    PushStack(s, s0[i]);
                    i++;
                }
            }
        }
        if (s.top == s.base)
            cout << "yes" << endl;
        else
        {
            cout << "no" << endl;
            switch (*--s.top)
            {
            case '{':cout << "{期待右括号" << endl;
                break;
            case '[':cout << "[期待右括号" << endl;
                break;
            case '(':cout << "(期待右括号" << endl;
                break;
            case '}':cout << "}期待左括号" << endl;
                break;
            case ']':cout << "]期待左括号" << endl;
                break;
            case ')':cout << ")期待左括号" << endl;
                break;
            }
        }
    }

}
int main()
{
    s.base = new char;
    s.top = s.base;
    char c, s0[100];
    int i = 0, k = 0;
    while ((c = getchar()) != EOF)
    {
        if (c == '(' || c == ')' || c == '{' || c == '}' || c == '[' ||
c == ']')
            s0[k++] = c;
    }
    s0[k] = '\0';
    Match(s, s0);
    return 0;
}
```

Problem-4

```
#include<iostream>
using namespace std;
#define MAX_SIZE 100
struct SeqStack
{
    char *top;
    char *base;
};
struct opdStack
{
    int top;
    int val[MAX_SIZE];
};
void IniStack(SeqStack &s)
{
    s.base = new char;
    s.top = s.base;
}
void PopStack(SeqStack &s)
{
    s.top--;
}
bool IsEmpty(SeqStack &s)
{
    if (s.top == s.base)
        return true;
    else
        return false;
}
void PushStack(SeqStack &s, char e)
{
    *s.top++ = e;
}
```

```
bool GetTop(SeqStack &s, char &e)
{
    if (IsEmpty(s))
        return false;
    e = *(s.top - 1);
    return true;
}
void PrintStack(SeqStack &s)
{
    while (s.top != s.base)
        cout << *--s.top;
}
void ConvertExp(SeqStack &s, char m[], char b[],int & flag)
{
    int i = 0, k = 0;
    char c, c1;
    c = m[i];
    while (c != '=')
    {
        if (c == '+' || c == '-')
        {
            while (!IsEmpty(s) && GetTop(s, c1) && c1 != '(')
            {
                PopStack(s);
                b[k++] = c1;
            }
            PushStack(s, c);
        }
        else if (c == '*' || c == '/')
        {
            while (!IsEmpty(s) && GetTop(s, c1) && (c1 == '*' || c1 ==
'/'))
            {
                PopStack(s);
                b[k++] = c1;
            }
            PushStack(s, c);
        }
        else if(c== '(')
            PushStack(s, c);
        else if (c == ')')
```

```
            {
                while (GetTop(s, c1) && c1 != '(')
                {
                    PopStack(s);
                    b[k++] = c1;
                }
                PopStack(s);
            }
            else if (c >= '0'&&c <= '9')
            {
                while (c >= '0'&&c <= '9')
                {
                    b[k++] = c;
                    c = m[++i];
                }
                i--;
                b[k++] = ' ';
            }
            else
            {
                flag = 1;
                return;
            }
            c = m[++i];
        }
        while (!IsEmpty(s))
        {
            GetTop(s, c1);
            PopStack(s);
            b[k++] = c1;
        }
        b[k] = '\0';
    }
    int Calculate(opdStack& opd, char b[])
    {
        int i = 0, value = 0, tmp = 0;
        int v1 = 0, v2 = 0;
        char c = b[i];
        while (c != '\0')
        {
            value = 0;
```

```
switch (c)
{
case '+':
    v2 = --opd.top;
    v1 = --opd.top;
    tmp = opd.val[v1] + opd.val[v2];
    opd.val[opd.top] = tmp;
    opd.top++;
    break;
case '-':
    v2 = --opd.top;
    v1 = --opd.top;
    tmp = opd.val[v1] - opd.val[v2];
    opd.val[opd.top] = tmp;
    opd.top++;
    break;
case '*':
    v2 = --opd.top;
    v1 = --opd.top;
    tmp = opd.val[v1] * opd.val[v2];
    opd.val[opd.top] = tmp;
    opd.top++;
    break;
case '/':
    v2 = --opd.top;
    v1 = --opd.top;
    if (opd.val[v2] == 0)
        return 0;
    tmp = opd.val[v1] / opd.val[v2];
    opd.val[opd.top] = tmp;
    opd.top++;
    break;
default:
    while (b[i] != ' ')
    {
        value = value * 10 + (b[i] - '0');
        i++;
    }
    opd.val[opd.top++] = value;
}
c = b[++i];
```

```
    }
    return  opd.val[--opd.top];
}
int main()
{
    SeqStack s;
    opdStack opd;
    int flag = 0;
    opd.top = 0;
    IniStack(s);
    char Mid_exp[MAX_SIZE], Back_exp[MAX_SIZE];
    cin >> Mid_exp;
    ConvertExp(s, Mid_exp, Back_exp, flag);
    if (flag == 1)
        cout << "ERROR" << endl;
    else
        if (Calculate(opd, Back_exp) == 0)
            cout << "ERROR" << endl;
        else
            cout << Calculate(opd, Back_exp) << endl;
    return 0;
}
```

Problem-5

方法一：

```
#include<iostream>
#include<cstring>
using namespace std;
#define MAX_SIZE 1000
#define OK      1
#define ERROR   0
#define LOVERFLOW   -2
typedef  int Status;
typedef int  SElemType;
#define  STACK_INIT_SIZE  100   //初始大小为100
#define  STACKINCREMENT  10   //若空间不够，每次增长10
class  SqStack
{
protected:
    SElemType *base;
    SElemType *top;
    int   stacksize;
public:
    SqStack();
    ~SqStack();
    Status ClearStack();
    Status GetTop(SElemType&e);
    Status Push(SElemType e);
    Status Pop(SElemType&e);
};
SqStack s;
SqStack::SqStack()
{
    base = new SElemType[STACK_INIT_SIZE];
    if (base == NULL)
        exit(LOVERFLOW);
    top = base;
    stacksize = STACK_INIT_SIZE;
}
SqStack::~SqStack()
{
    if (base)
        delete base;
    top = NULL;
    stacksize = 0;
}
Status SqStack::ClearStack()
{
    if (stacksize > STACK_INIT_SIZE)
    {/*如果栈扩展过，恢复初始大小*/
        delete base;
        base = new SElemType[STACK_INIT_SIZE];
        if (base == NULL)
            exit(LOVERFLOW);
        stacksize = STACK_INIT_SIZE;
    }
    top = base;
    return OK;
}
Status SqStack::GetTop(SElemType &e)
```

```
{
    if (top == base)
        return ERROR;
    e = *(top - 1);
    return OK;
}
Status SqStack::Push(SElemType e)
{
    /*如果栈已满则扩充空间*/
    if (top - base >= stacksize)
    {
        SElemType *newbase;
        newbase = new SElemType[stacksize + STACK_INIT_SIZE];
        if (!newbase)
            return LOVERFLOW;
        memcpy(newbase, base, stacksize * sizeof(SElemType));
        delete base;
        base = newbase;
        top = base + stacksize;
        stacksize += STACKINCREMENT;
    }
    *top++ = e;
    return OK;
}
Status SqStack::Pop(SElemType &e)
{
    if (top == base)
        return ERROR;
    e = *--top;
    return OK;
}
Status Judge_OK(SqStack &s, char Odr[], char InOdr[], int iLen)
{
    int i = 0, j = 0;
    SElemType e;
    while (1)
    {
        if (InOdr[i] == Odr[j])
        {
            i++;
            j++;
            if (j == iLen)
                return OK;
        }
        else if (s.GetTop(e) && e == Odr[j])
        {
            s.Pop(e);
            j++;
            if (j == iLen)
                return OK;
        }
        else
        {
            if (i == iLen)
                return ERROR;
            s.Push(InOdr[i]);
            i++;
        }
    }
}
int main()
{
    char InOdr[10];
    char OutOdr[MAX_SIZE][10];
    int iLen = 0, oLen = 0;
    cin.getline(InOdr, 10);
    int i;
    for (i = 0; i < 1000; i++)
    {
        if (!cin.getline(OutOdr[i], 10))
            break;
    }
    iLen = strlen(InOdr);
    for (int k = 0; k < i; k++)
    {
        s.ClearStack();
        oLen = strlen(OutOdr[k]);
        if (oLen != iLen)
            cout << "no" << endl;
        else
        {
            if (Judge_OK(s, OutOdr[k], InOdr, iLen))
                cout << "yes" << endl;
            else
                cout << "no" << endl;
        }
    }
    return 0;
}
```

方法二：

```
#include<iostream>
#include<string>
using namespace std;
#define MAX_SIZE 1000
struct InOdr
{
    char Iorder;
    int iNo;
    int iLen;
};
InOdr odr[MAX_SIZE];
InOdr odra;
struct OutOdr
{
    char Oorder;
    int oNo;
    int oLen;
};
OutOdr str[MAX_SIZE][10];
struct SeqStack
{
    InOdr *top;
    InOdr *base;
};
SeqStack  s;
void IniStack(SeqStack &s)
{
    s.base = &odra;
    s.top = s.base;
}
void PopStack(SeqStack &s, InOdr &e)
```

```
{
    s.top--;
    e = *s.top;
}
void PushStack(SeqStack &s, InOdr& e)
{
    *s.top++ = e;
}
bool Judge_OK(SeqStack &s, OutOdr str[][10], InOdr odr[], int i)
{
    int  j = 1, k = 1;
    int count = 0, sum = 1;
    InOdr c;
    if (str[i][k].oNo != 0)
    {
        while (sum <= str[i][k].oNo - count)
        {
            c = odr[j];
            PushStack(s, c);
            sum++;
            j++;
        }
        sum--;
        count += sum;
        sum = 1;
        PopStack(s, c);
    }
    else
        return false;

    while (c.Iorder == str[i][k].Oorder&&str[i][k].Oorder != '\0')
    {
        k++;
        if (str[i][k].oNo == 0 && str[i][k].Oorder != '\0')
            return false;
        else
        {
            if (str[i][k].Oorder == '\0')
                return true;
            if (str[i][k].oNo < c.iNo)
            {
                PopStack(s, c);
            }
            else  if (str[i][k].oNo > c.iNo)
            {
                while (sum <= str[i][k].oNo - count)
                {
                    c = odr[j];
                    PushStack(s, c);
                    sum++;
                    j++;
                }
                sum--;
                count += sum;
                sum = 1;
                PopStack(s, c);
            }
        }
    }
    if (c.Iorder != str[i][k].Oorder)
        return false;
}
void Assign_No(InOdr odr[], OutOdr str[][10], int i, int j, int vis[])
{
    int k = 1;
    while (odr[k].Iorder != '\0')
    {
        if (odr[k].Iorder == str[i][j].Oorder&&vis[odr[k].iNo] == 0)
        {
            str[i][j].oNo = odr[k].iNo;
            vis[odr[k].iNo] = 1;
            break;
        }
        k++;
    }
}
int main()
{
    int i = 1;
    char c0;
    string ans[MAX_SIZE];
    int vis[MAX_SIZE] = { 0 };
```

```
IniStack(s);
while ((c0 = getchar()) != '\n')
{
    odr[i].iNo = i;
    odr[i].Iorder = c0;
    i++;
}
odr[i].iLen = i - 1;
odr[i].Iorder = '\0';
char c;
int k = 1, j = 1;
while ((c = getchar()) != EOF)
{
    if (c != '\n')
    {
        str[k][j].Oorder = c;
        Assign_No(odr, str, k, j, vis);
        j++;
    }
    else
    {
        str[k][j].Oorder = '\0';
        str[k][j].oLen = j - 1;
        if (str[k][j].oLen == odr[i].iLen)
            if (Judge_OK(s, str, odr, k))
                ans[k] = "yes";
            else
                ans[k] = "no";
        else
            ans[k] = "no";
        k++;
        for (j = 1; j <= odr[i].iLen; j++)
            vis[j] = 0;
        j = 1;
    }
}
ans[k] = " ";
while (ans[j] != " ")
    cout << ans[j++] << endl;
system("pause");
return 0;}
```