题意

规定下列表达式为合法表达式

- d 是合法的表达式，当 d 为一位正整数
- （a op b) 是合法的表达式，当a , b 也为合法的表达式，op为操作符

操作符只有+ ， -

给出一个合法的表达式，所有的操作符用 '?' 代替

给出该表达式中 + 号数目 p, 减号数目 m,保证 p+m == 表达式中操作符数目 min(p,m) <= 100

求该表达式最大结果

分析

将表达式建成一棵树

- 每个叶子结点为一个数字
- 每个非叶子结点为一个操作符，左右子树为操作的表达式

建树方法：从（0,s.size()-1) 向下递归建树 build(L,R)

- 如果L == R ,则此处为叶子结点
- 如果L< R 遍历该区间，维护数字k（初始为0）遇到左括号 +1,遇到右括号 -1 ， 当遇到操作符，并且此时k == 1(左边还剩一个未配对左括号） ，从该操作符所在位置x为界，将表达式分为左右两个表达式，去掉两端的括号，递归求解build(L+1,x-1),build(x+1,R-1)

```
int build(int L,int R)
{
    if(L == R)
    {
        tree[++tot] = {0,0,0};
        Max[tot][0] = s[L] - '0';
        Min[tot][0] = s[R] - '0';
        return tot;
    }
    int k = 0;
    For(i,L,R)
    {
        if(s[i] == '(') k++;
        else if(s[i] == ')') k--;
        else if(s[i] == '?' && k == 1)
```

```
16          {
17              int l = build(L+1,i-1),r = build(i+1,R-1);
18              tree[++tot] = Node{tree[l].siz + tree[r].siz + 1,l,r};
19              int k = min(p,m);
20              return tot;
21          }
22      }
23  }
```

子树最大值 = 左子树最大值 + 右子树最大值
= 左子树最大值 - 右子树最小值
子树最小值 = 左子树最小值 + 右子树最大值
= 左子树最小值 - 右子树最大值
所以可以自底向上，求得子树的最小值和最大值，最后的答案就为根节点的最大值
考虑到min（p,m) <= 100，最多有20000个结点，设数量小的操作符为op1，可以枚举求得该子树op1不超过 子树操作符数目以及给定op1最大值的最大值和最小值，由于两种情况有差异，可以分类讨论

```
1   void dfs(int now)
2   {
3       int l = tree[now].lc,r = tree[now].rc;
4       if(l == r ) return;
5       dfs(l);dfs(r);
6       if(p <= m)
7       {
8           For(i,0,p)  For(j,0,i)
9               if(tree[l].siz >= j && tree[r].siz >= i-j){
10                  Max[now][i] = max(Max[now][i],Max[l][j] - Min[r][i-j]);
11                  Min[now][i] = min(Min[now][i],Min[l][j] - Max[r][i-j]);
12              }
13          For(i,0,p)  For(j,0,i-1)
14              if(tree[l].siz >= j && tree[r].siz >= i-j-1){
15                  Max[now][i] = max(Max[now][i],Max[l][j]+Max[r][i-j-1]);
16                  Min[now][i] = min(Min[now][i],Min[l][j]+Min[r][i-j-1]);
17              }
18      }
19      else
20      {
21          For(i,0,m)  For(j,0,i)
22              if(tree[l].siz >=j && tree[r].siz>= i-j){
23                      Max[now][i] = max(Max[now][i],Max[l][j]+Max[r][i-j]);
24                      Min[now][i] = min(Min[now][i],Min[l][j]+Min[r][i-j]);
25              }
26          For(i,0,m)  For(j,0,i-1)
```

```
            if(tree[l].siz >= j && tree[r].siz >= i-j-1){
                    Max[now][i] = max(Max[now][i],Max[l][j] - Min[r][i-j-1]);
                    Min[now][i] = min(Min[now][i],Min[l][j] - Max[r][i-j-1]);
            }
    }
}
```

代码

```
#include<algorithm>
#include<bitset>
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<deque>
#include<iostream>
#include<map>
#include<queue>
#include<set>
#include<stack>
#include<string>
#include<vector>
#include<list>
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
const int INF = 0x3f3f3f3f;
typedef  long long LL;typedef  unsigned long long ULL;typedef  long double LDB;
inline LL CinLL(){LL x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}
inline int Cin(){int x=0,f=1;char ch=getchar();while(!isdigit(ch)){if(ch=='-')f=-1;ch=getchar();}while(isdigit(ch))x=x*10+ch-'0',ch=getchar();return f*x;}
const int N = 1e5+5;
struct Node{
    int siz,lc,rc;
}tree[2*N];
```

```
int tot = 0;
int Max[2*N][105],Min[2*N][105];
string s;
int p,m;
int build(int L,int R)
{
    if(L == R)
    {
        tree[++tot] = {0,0,0};
        Max[tot][0] = s[L] - '0';
        Min[tot][0] = s[R] - '0';
        return tot;
    }
    int k = 0;
    For(i,L,R)
    {
        if(s[i] == '(') k++;
        else if(s[i] == ')') k--;
        else if(s[i] == '?' && k == 1)
        {
            int l = build(L+1,i-1),r = build(i+1,R-1);
            tree[++tot] = Node{tree[l].siz + tree[r].siz + 1,l,r};
            int k = min(p,m);
            return tot;
        }
    }
}
void dfs(int now)
{
    int l = tree[now].lc,r = tree[now].rc;
    if(l == r ) return;
    dfs(l);dfs(r);
    if(p <= m)
    {
        For(i,0,p)  For(j,0,i)
            if(tree[l].siz >= j && tree[r].siz >= i-j){
                Max[now][i] = max(Max[now][i],Max[l][j] - Min[r][i-j]);
                Min[now][i] = min(Min[now][i],Min[l][j] - Max[r][i-j]);
            }
        For(i,0,p)  For(j,0,i-1)
            if(tree[l].siz >= j && tree[r].siz >= i-j-1){
                Max[now][i] = max(Max[now][i],Max[l][j]+Max[r][i-j-1]);
                Min[now][i] = min(Min[now][i],Min[l][j]+Min[r][i-j-1]);
            }
    }
    else
```

```
77      {
78          For(i,0,m)  For(j,0,i)
79              if(tree[l].siz >=j && tree[r].siz>= i-j){
80                      Max[now][i] = max(Max[now][i],Max[l][j]+Max[r][i-j]);
81                      Min[now][i] = min(Min[now][i],Min[l][j]+Min[r][i-j]);
82              }
83          For(i,0,m)  For(j,0,i-1)
84              if(tree[l].siz >= j && tree[r].siz >= i-j-1){
85                      Max[now][i] = max(Max[now][i],Max[l][j] - Min[r][i-j-1]);
86                      Min[now][i] = min(Min[now][i],Min[l][j] - Max[r][i-j-1]);
87              }
88      }
89  }
90  int main()
91  {
92      ios::sync_with_stdio(false);
93      memset(Max,0x8f,sizeof Max);
94      memset(Min,0x3f,sizeof Min);
95      cin>>s;
96      int root = build(0,s.size()-1);
97      cin>>p>>m;
98      dfs(root);
99      cout<<Max[root][min(p,m)]<<endl;
100 }
```