第三章作业答案

3.6

解：因为 Pj<Pk<Pi，所以在输入时 Pj 和 Pk 肯定比 Pi 先入栈，即 Pi 最后入栈；又因为 Pi 比 Pj 和 Pk 先出栈，所以在 Pi 出栈时，Pj 和 Pk 一定还在栈中；又因为 Pj 小于 Pk，所以 Pj 一定在 Pk 的下面，则出栈时 Pk 一定比 Pj 先出栈，所以不可能存在 i<j<k 的情况。

（注意：对于该题，有些同学对题意有所误解，该题中的在入栈时也可以进行出栈操作，并不是将所有元素都入栈了才可以出栈）

3.19

此题易错点：

在遇到右括号时，应先判断栈中是否还有元素，再进行出栈操作；

在完成匹配之后，应判断栈中是否还有元素。

```
bool problem3_19(char* str)
{
    Stack S;
    InitStack(S);
    char* c = str;
    char e;
    //假设输入字符串以'#'作为结束字符
    while (*c != '#')
    {
        if (*c == '(' || *c == '[' || *c == '{')
            Push(S, *c);
        else if (*c == ')' || *c == ']' || *c == '}')
        {
            //如果此时没有左括号可以匹配，则表达式不正确
            if (StackEmpty(S))
                return false;
            Pop(S, e);
            if ((e == '('&&*c != ')') || (e == '['&&*c != ']') || (e
== '{'&&*c != '}'))
                return false;
        }
        c++;
    }
    //最后应判断栈是否为空，因为若栈中还有剩余左括号未被匹配，说明表达式不正确
    return StackEmpty(S);
}
```

3.21

```
//输入字符串以'#'作为结束字符
char* problem3_21(char* str)
{
    Stack S;
    InitStack(S);
    char* c = str, e;
    char* buffer = new char[length(str)];
    int i = 0;
    Push(S, *c);
    c++;
    while (*c != '#')
    {
        if ((*c >= 'a' && *c <= 'z')|| (*c >= 'A' && *c <= 'Z'))
        {
            buffer[i++] = *c;
        }
        else if (*c == '*' || *c == '/' || *c == '+' || *c == '-')
        {
            GetTop(S, e);
            //当栈顶操作符优先权高于当前优先符时，出栈
            if (Prior(e, *c))
            {
                Pop(S, e);
                buffer[i++] = e;
                continue;
            }
            else
                Push(S, *c);

        }
        c++
    }
    //最后应该将栈中剩余的元素弹出
    while (!StackEmpty(S))
    {
        Pop(S, e);
        buffer[i++] = e;
    }
    return buffer;
}
```

3.27（1）

```
int akm(int m, int n)
{
    if (m == 0)
        return n + 1;
    else if (n == 0)
        return akm(m - 1, 1);
    else
        return akm(m - 1, akm(m, n - 1))
}
```

3.31

```
bool problem3_31(char* str)
{
    Stack S;
    Queue Q;
    InitStack(S);
    InitQueue(Q);
    char* p = str;
    char a, b;
    while (*p != '@')
    {
        Push(S, *p);
        EnQueue(Q, *p);
        p++;
    }
    while (!StackEmpty(S))
    {
        Pop(S, a);
        DeQueue(Q, b);
        if (a != b)
            return false;
    }
    return true;
}
```

3.32

该题主要错误点：部分同学将其作为 2 阶的斐波那契数列进行计算。K 阶斐波那契数列：数列第 1 项到第 k-1 项为 0，第 k 项为 1，之后从第(k+1)项开始每一项为前 k 项之和。

```
void problem3_32(int k, int max)
{
    Queue Q;
```

```
    InitQueue(Q, k);
    int i = 0, e;
    int sum = 0;
    //前k-1项全为0
    for (i = 0; i < k - 1; i++)
        EnQueue(Q, 0);
    //第k项为1
    EnQueue(Q, 1);
    while (sum<=max)
    {
        sum = 0;
        //因为该循环队列最多可以存k项，所以对整个队列元素求和，就可得到斐波那契数列下一项的值
        for (i = 0; i < k; i++)
            sum += Q.base[i];
        Q.rear = (Q.rear + 1) % k;
        Q.base[Q.rear] = sum;
        Q.front = (Q.front + 1) % k;

    }

    //如下代码也可以
    /*
    while (sum <= max)
    {
        sum = 0;
        for (i = 0; i < k; i++)
            sum += Q.base[i];
        DeQueue(Q, e);
        EnQueue(Q, sum);
    }
    */
}
```