数据结构 A

第三章参考答案

题目范围：栈与队列

授课老师：何璐璐

助　　教：赵守玺(撰写人)、王思怡

联系邮箱：sxzhao@whu.edu.cn

目录

1. 作业一

1.1 编程题

1.1.1 逆波兰表达式求值

【问题描述】

根据逆波兰表示法，求表达式的值。有效的运算符包括+，-，*，/。每个运算对象可以是整数，也可以是另一个逆波兰表达式。假设给定逆波兰表达式总是有效的，换句话说，表达式总会得出有效数值且不存在除数为 0 的情况。其中整数除法只保留整数部分。

【输入形式】

每个样例是一行，为有效的表达式，每个数字和运算符号之间用“,”隔开

【输出形式】

表达式的计算结果

【样例输入】

2,1,+,3,*

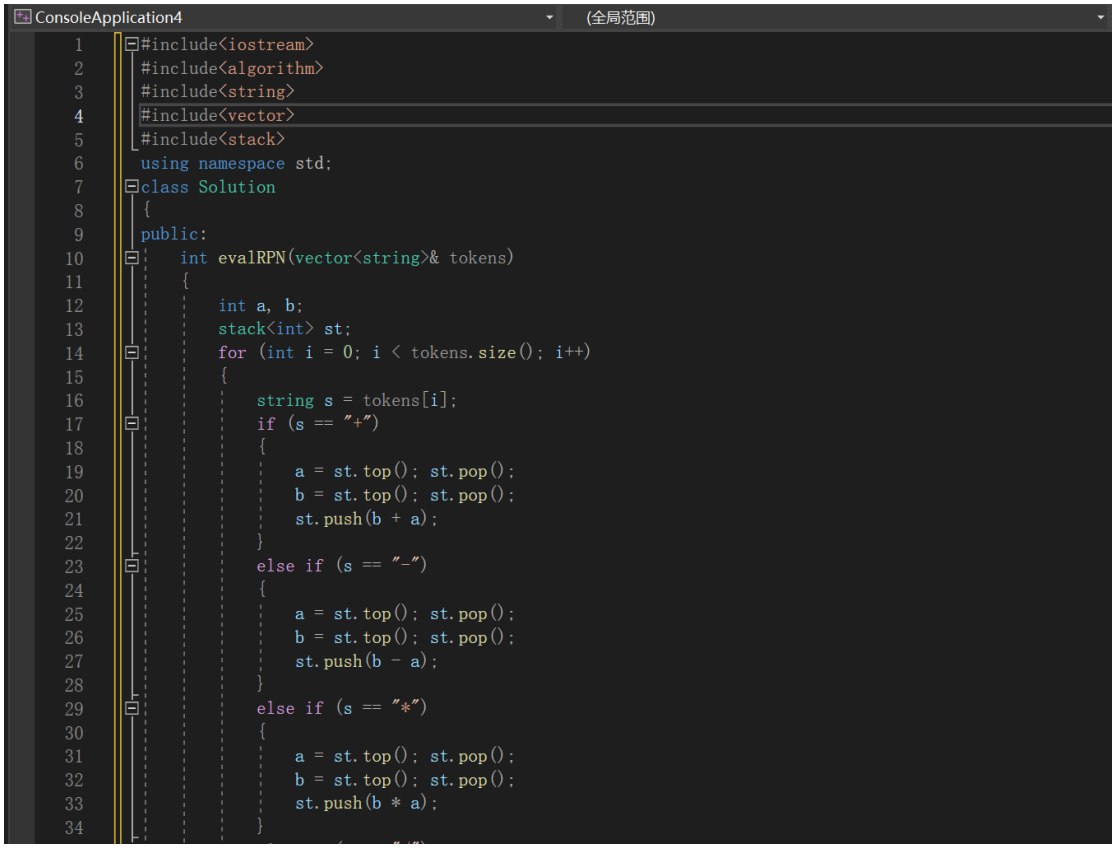
【样例输出】

9

【样例说明】

测试数据的文件名为 in.txt

参考答案：



解析：逆波兰表达式的求值考察的是栈的简单使用，我们可

以使用栈来跟踪操作数。遍历表达式，遇到数字时将其压入栈中，遇到运算符时弹出栈顶的两个数字，执行相应的运算，然后将结果压回栈中。最终，栈中的唯一元素就是表达式的值。具体实现时，可以按照以下步骤进行：

(1)初始化一个空栈。

(2)遍历逆波兰表达式中的每个元素：

(3)如果是数字，将其转换为整数并压入栈中。

(4)如果是运算符，弹出栈顶的两个数字，执行相应的运算，然后将结果压回栈中。

(5)返回栈中的唯一元素作为表达式的值。

1.1.2 合并栈操作

【问题描述】

栈是一种具有后进先出的数据结构。可合并栈是支持“merge”操作的栈。三种操作的说明如下：

① push A x：将 x 插入栈 A 中。

② pop A：删除栈 A 的顶部元素。

③ merge A B：合并栈 A 和 B。

其中，“merge A B”操作后栈 A 包含 A 和 B 之前的所有元素，B 变为空，新栈中的元素根据先前的进栈时间重新排列，就像在一个栈中重复"push"操作一样。给定两个可合并栈 A 和 B，请执行上述操作。

【输入形式】

测试用例的第一行包含一个整数 n（0<n≤10^5）表示操作个数，接下来的 n 行每行包含一条指令 push、pop 或 merge，栈元素是 32 位整数。A 和 B 最初都是空的，并且保证不会对空栈执行 pop 操作。以 n=0 表示输入结束。

【输出形式】

对于每个 pop 操作，在一行中输出对应的出栈元素。

【样例输入】

```
9
push A 0
push A 1
push B 3
pop A
push A 2
merge A B
pop A
pop A
pop A
```

【样例输出】

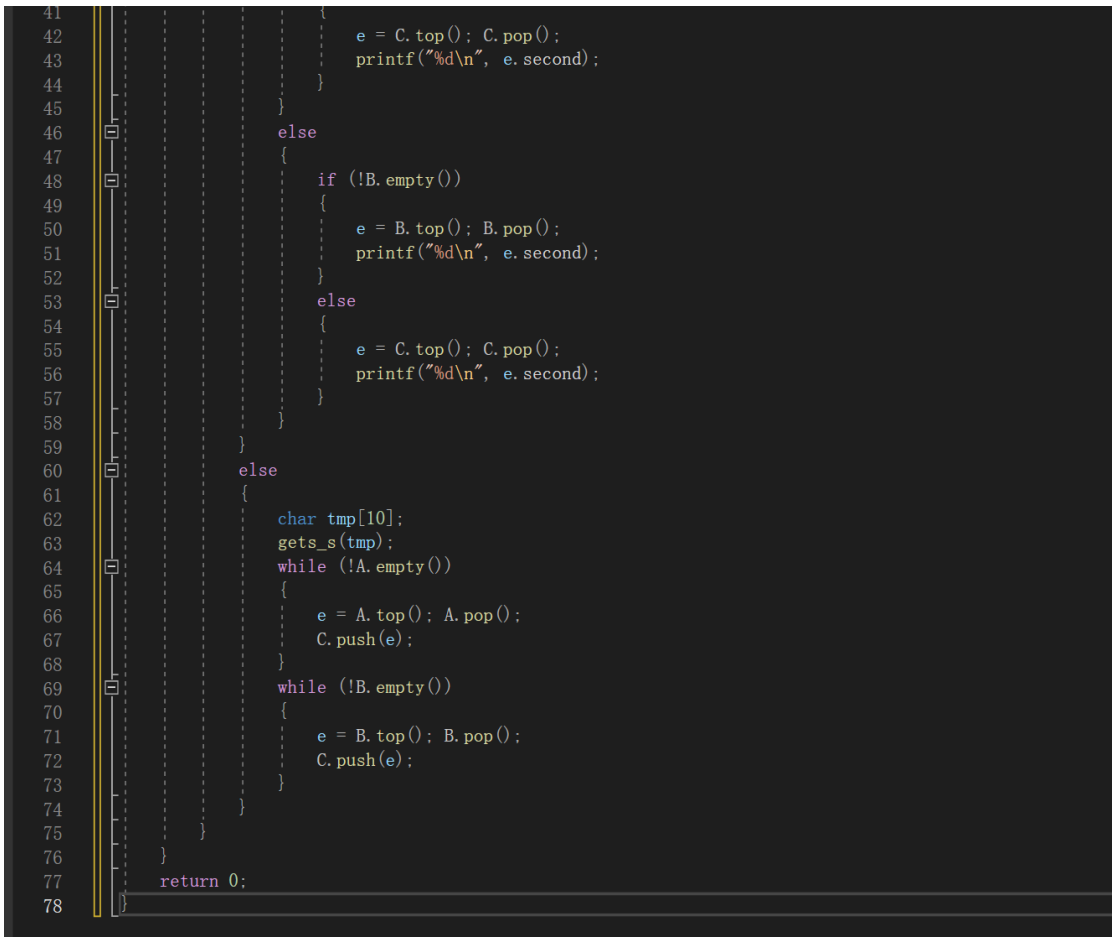
```
1
2
3
0
```

【样例说明】

测试数据的文件名为 in.txt

参考答案：

```cpp
#include<iostream>
#include<queue>
#include <stdio.h>
using namespace std;
typedef pair<int, int> PA;
priority_queue<PA> A, B, C;
int main()
{
    freopen("in.txt", "r", stdin);
    int n, x;
    int i = 1;
    PA e;
    char com[10], g;
    while (scanf("%d", &n) != EOF && n)
    {
        while (!A.empty()) A.pop();
        while (!B.empty()) B.pop();
        while (!C.empty()) C.pop();
        int timeid = 0;
        while (n--)
        {
            scanf("%s %c", com, &g);
            if (com[1] == 'u')
            {
                scanf("%d", &x);
                if (g == 'A')
                    A.push(make_pair(timeid++, x));
                else
                    B.push(make_pair(timeid++, x));
            }
            else if (com[1] == 'o')
            {
                if (g == 'A')
                {
                    if (!A.empty())
                    {
                        e = A.top(); A.pop();
                        printf("%d\n", e.second);
                    }
                    else
                    {
```



```
                {
                    e = C.top(); C.pop();
                    printf("%d\n", e.second);
                }
            }
            else
            {
                if (!B.empty())
                {
                    e = B.top(); B.pop();
                    printf("%d\n", e.second);
                }
                else
                {
                    e = C.top(); C.pop();
                    printf("%d\n", e.second);
                }
            }
        }
        else
        {
            char tmp[10];
            gets_s(tmp);
            while (!A.empty())
            {
                e = A.top(); A.pop();
                C.push(e);
            }
            while (!B.empty())
            {
                e = B.top(); B.pop();
                C.push(e);
            }
        }
    }
    return 0;
}
```

解析：当解决这道题目时，我们需要维护两个栈 A 和 B，并根据操作指令进行相应的处理。以下是解题思路：

(1)初始化两个空栈 A 和 B。

(2)遍历输入的操作指令：

如果是"push A x"，将元素 x 压入栈 A。

如果是"pop A"，弹出栈 A 的顶部元素并输出。

如果是"merge A B"，将栈 B 的元素依次弹出并压入栈 A，保持先前的进栈时间顺序。

(3)重复步骤 2 直到处理完所有操作指令。

2. 作业二

2.1 单选题

1、设一个栈的输入序列是 a,b,c,d，则借助一个栈所得到的输出序列不可能是 _____

A.a,b,c,d
B.d,c,b,a
C.a,c,d,b
D.d,a,b,c

答案：D，d 最后一个入栈且第一个出栈，就说明其余三个元素在 d 之前没有出栈。d 出栈后只能是 c 出栈

2、已知一个栈的进栈序列是 1，2，3，…，n，其输出序列是 p1，p2，…，pn，若 p1=n，则 pi 的值是 _____

A.i
B.n-i+1
C.n-i
D.不确定

答案：B，我们可以观察到：元素 1 是第一个进栈的，而在输出序列中，它是最后一个出栈的；元素 2 是第二个进栈的，而在输出序列中，它是倒数第二个出栈的。依此类推，进栈序列中的元素 i 是第 i 个进栈的，而在输出序列中，它是倒数第 i 个出栈的。

3、一个栈的入栈序列为 1，2，3，…，n，其出栈序列为 p1，p2，…，pn，若 p2=3，则 p3 可能取值的个数是 _____

A.n-1
B.n-2
C.n-3
D.无法确定

答案：A，由于 p2 是 3，那么 p1 只可能是 1 或 2，无论是哪个。都使得 p3 不能是前两个已经出现过的数字，故 p3 的情况总数为 n+1-2=n-1。

4、由两个栈共享一个数组空间的好处是 _____
A.减少存取时间，降低上溢出发生的几率
B.节省存储空间，降低上溢出发生的几率
C.减少存取时间，降低下溢出发生的几率
D.节省存储空间，降低下溢出发生的几率

答案：B，当我们需要使用两个栈时，每个栈都需要一个独立的数组来存储元素。如果我们共享一个数组空间，那么这两个栈可以共用同一块内存，从而减少了内存的占用。

5、表达式 a*（b+c）-d 的后缀表达式是 _____
A.abcd*+-
B.abc+*d-
C.abc*+d-
D.-+*abcd

答案：B，按照书上给的流程进行转换即可。

6、在算数表达式：1+6/（8-5）*3 转化成后缀表达式的过程中，当扫描到 5 时运算符栈（从栈顶到栈底次序）为 _____
A.-/+
B.-(/+
C./+
D./-+

答案：B，自己理一遍过程即可。

7、以下数据结构中元素之间为线性关系的是 _____
A.栈
B.队列
C.线性表
D.以上都是

答案：D，线性结构满足以下特征：(1)集合中必存在唯一的一个“第一个元素”。(2)集合中必存在唯一的一个“最后的元素”。(3)除最后元素之外，其它数据元素均有唯一的“后继”。(4)除第一元素之外，其它数据元素均有唯一的“前驱”。

8、以下各链表均不带头结点，其中最不适合用作链栈的链表是 _____
A.只有表头指针没有表尾指针的循环双链表
B.只有表尾指针没有表头指针的循环双链表
C.只有表头指针没有表尾指针的循环单链表
D.只有表尾指针没有表头指针的循环单链表

答案：C，栈这种数据结构必须遵从“后进先出”的数据存取规律。链表是在尾部加元素，所以就需要从尾部取元素。而表头是从头开始取元素，所以不适合上述规律。

9、环形队列 _____
A.不会产生下溢出
B.不会产生上溢出
C.不会产生假溢出
D.以上都不对

答案：C，因为在数组里，头尾指针只增加不减少，被删元素的空间再也不能被重新利用。会造成尾指针已经到达了队列的最后一位，而头指针前面没有满的情况。

10、已知环形队列存储在一维数组 A[0..n-1]中，且队列非空时 front 和 rear 分别指向队头元素和队尾元素。若初始时队列空，且要求第一个进入队列的元素存储在 A[0]处，则初始时 front 和 rear 的值分别是 _____

A.0，0

B.0，n-1

C.n-1，0

D.n-1，n-1

答案：B，在循环队列中,进队操作是队尾指针 rear 循环加 1,再在该处放置进队的元素,本题要求第一个进入队列的元素存储在 A[0]处,则 rear 应为 n-1,因为这样(rear+1)%n=0。而队头指向队头元素,此时队头位置为 0,所以 front 的初值为 0。

11、设环形队列的存储空间是 a【0..20】，且当前队头指针（f 指向队首元素的前一个位置）和队尾指针（r 指向队尾元素）的值分别是 8 和 3，则该队列中的元素个数为 _____

A.5

B.6

C.16

D.17

答案：C，这里 MaxSize=21,其中的元素个数=(r-f+MaxSize)%MaxSize=16。

12、最适合用作链队的链表是 _____
A.带队首指针和队尾指针的循环单链表
B.带队首指针和队尾指针的非循环单链表
C.只带队首指针的非循环单链表
D.只带队首指针的循环单链表

答案：B，队列既需要访问队头也需要访问队尾。

2.2 多选题

1、假如栈的入栈顺序是 a,b,c,d，下面 4 个选项中可能是它的出栈顺序的是 _____
A. a,c,b,d
B. b,c,d,a
C. c,d,b,a
D. d,c,a,b

答案：ABC，在这种情况下 b 一定比 a 先出栈。

2.3 填空题

1、把中缀表达式 3+（5-2）*6 转化为后缀表达式时，需要的顺序栈容量至少是 _____ ，得到的后缀表达式是 _____ （用#表示一个数字串结束）。

答案：3 ， 3#5#2#-6#*+。

2、一个队列的入队序列是 1 2 3 4 5 6 7，则可能的出队序列有 _____ 个，它（们）是: _____

答案：1，1 2 3 4 5 6 7。

2.4 编程题

2.4.1 团队队列

【问题描述】

在团队队列中每个成员都属于一个团队，如果一个成员进入队列，它首先从头到尾搜索队列，以检查它的一些队友（同一队的成员）是否已经在队列中，如果是，它会进入到该团队的后面，如果不是，它会从尾部进入队列并成为新的最后一个成员。成员出队是按常规队列操作，按照出现在队列中的顺序从头到尾进行处理。你的任务是编写一个模拟这样的团队队列的程序。

【输入形式】

每个测试用例都以团队个数 t 开始（1≤t≤1000），然后是团队描述，每个描述包含属于团队的成员个数和成员编号列表，成员编号为 0 到 999999 之间的整数，一个团队最多可以包含 1000 个成员。然后是一系列命令，有三种不同的命令：

① ENQUEUE p：成员 p 进入队列。

② DEQUEUE：队列中第一个成员出来并将其从队列中删除。

③ STOP：当前测试用例结束。

【输出形式】

对于每个 DEQUEUE 命令，以单独一行输出出队的成员。

【样例输入】

```
2
3 101 102 103
3 201 202 203
ENQUEUE 101
ENQUEUE 201
ENQUEUE 102
ENQUEUE 202
ENQUEUE 103
ENQUEUE 203
DEQUEUE
DEQUEUE
DEQUEUE
DEQUEUE
DEQUEUE
DEQUEUE
STOP
```

【样例输出】

```
101
102
103
201
202
203
```

【样例说明】

测试数据的文件名为 in.txt

答　　案　　：

```cpp
#include<iostream>
#include<queue>
#include<string>
#include<cstring>
using namespace std;

const int MAXN = 1000000;
int termno[MAXN];
int n;
queue<int> totalqu;
struct
{
    queue<int> qu;
    bool vis;
} termqu[1001];
void solve(int cas)
{
    string com;
    while (cin >> com && com != "STOP")
    {
        if (com == "ENQUEUE")
        {
            int p;
            cin >> p;
            termqu[termno[p]].qu.push(p);
            if (!termqu[termno[p]].vis)
            {
                totalqu.push(termno[p]);
                termqu[termno[p]].vis = true;
            }
        }
        else if (com == "DEQUEUE")
        {
            cout << termqu[totalqu.front()].qu.front() << endl;
            termqu[totalqu.front()].qu.pop();
            if (termqu[totalqu.front()].qu.empty())             {
                termqu[totalqu.front()].vis = false;
                totalqu.pop();
            }
        }
        else break;
    }
```

```
}
void init()
{
    memset(termno, 0, sizeof(termno));
    while (!totalqu.empty())
        totalqu.pop();
    for (int i = 1; i <= n; i++)
    {
        while (!termqu[i].qu.empty())
            termqu[i].qu.pop();
        termqu[i].vis = false;
    }
}
int main()
{
    int cas = 1;
    int t, p;
    if (freopen("in.txt", "r", stdin) == NULL)
    {
        cout << "??????????";
        return 1;
    }
    cin >> n;
    init();
    for (int i = 1; i <= n; i++)
    {
        cin >> t;
        for (int j = 1; j <= t; j++)
        {
            cin >> p;
            termno[p] = i;
        }
    }
    solve(cas++);
    cout << endl;
    return 0;
}
```

解析： 看题目,如果我们按照题目,用一个队列来维护整个Team Queue,用 STL 则无法实现,而手写也不是非常容易实现.所以,我们要想一个更简单的办法:既然题目说这里面有许多不同的团队,那么我们为什么不用一个大的队列只记录团队号的顺序,再用一些小的队列来记录每个团队内元素的先后顺序.每当读入元素,我们判断它所在的团队是否在大序列中,如果在,则直接将这个元素插入到它的团队队列中;如果不在,则将它的团队号插入到大队列中,再将它插入到它的团队队列中.每当要输出元素,便找大队列中最前面的团队的第一个元素,弹出,如果这个团队的序列空了,则将这个团队的团队号从大队列中弹出去.

2.4.2 扔钉子

【问题描述】

年度学校自行车比赛开始了，ZL 是这所学校的学生，他太无聊了，因为他不能骑自行车！因此，他决定干预比赛，他通过以前的比赛视频获得了选手的信息，一个选手第一秒可以跑 F 米，然后每秒跑 S 米。每个选手有一条直线跑道，ZL 每秒向跑的最远的运动员跑道扔一个钉子，在自行车胎爆炸之后，该选手将被淘汰。如果有多个选手是 NO.1，则他总是选择 ID 最小的选手扔钉子。

【输入形式】

每个测试用例的第一行包含一个整数 n（1≤n≤50000），表示选手人数，然后跟随 n 行，每行包含第 i 个选手的两个整数 Fi（0≤Fi≤500），Si（0<Si≤100），表示该选手第一秒可以跑 Fi 米，然后每秒跑 Si 米，i 是玩家从 1 开始的 ID。

【输出形式】

输出 n 个数字，以空格分隔，第 i 个数字是选手的 ID，该选手将在第 i 秒结束时被淘汰。

【样例输入】

```
3
100 1
100 2
3 100
```

【样例输出】

```
1 3 2
```

【样例说明】

测试数据的文件名为 in.txt

答　　案　　：

```cpp
#include<iostream>
#include<algorithm>
#include<queue>

using namespace std;
struct QNode
{
    int id;
    int len;
    bool operator < (const QNode& s)const
    {
        if (len == s.len)
            return id > s.id;
        else
            return len < s.len;
    }
};
priority_queue<QNode> qu[105];
int main()
{
    freopen("in.txt", "r", stdin);
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        int f, s;
        QNode p;
        cin >> f >> s;
        p.len = f;
        p.id = i + 1;
        qu[s].push(p);
    }
    for (int i = 0; i < n; i++)
    {
        int pos, pid;
        int maxs = -1;
        for (int j = 1; j < 101; j++)
        {
            if (!qu[j].empty())
            {
                QNode p = qu[j].top();
                if (p.len + j * i > maxs || (p.len + j * i == maxs && p.id < pid))
                {
                    pos = j;
                    pid = p.id;
                    maxs = p.len + j * i;
                }
            }
        }
        if (i != n - 1)
            cout << pid << " ";
        else
            cout << pid;
        qu[pos].pop();
    }
    cout << endl;
}
```

解析：这道题目的解题思路如下：

Si 最大只有 100，可建立优先队列数组 s[1..100]，对于每个优先队列，按第一关键字 Fi 第二关键字 ID 排序，每次取出所有的优先队列里最大值，然后直接 计算（Time-1）*Si + Fi 找最大的 way 将对应的优先队列 pop 并输出对应 ID 即可。