Информация по лабораторной

- В этой лабораторной использовать стандартную библиотеку (vector, ArrayList, LinkedList, и т.п.) не разрешается.
- В лабораторных рекомендуется использовать язык С++, а в этой лабораторной язык С.
- У компилятора C++ в системе жесткая проверка на предупреждения, она нужна, чтобы мы все научились следовать стандарту и не писать код, который может привести к непредвиденным ошибкам.
- Решение во всех задачах нужно реализовывать так, чтобы они работали при любом размере входных данных. Не везде это удобно делать, но надо пытаться это делать, чтобы код был более общий. (заводить массивы константного размера не разрешается)
- В этой лабораторной нужно следить за памятью, то есть вектор должен расширяемым и сужаемым, при добавлении и удалении элементов, соответственно. Структура данных должна занимать $\Theta(n)$ памяти, где n—число добавленных элементов.
- Во всех задачах работает ввод из stdin и вывод в stdout.
- В некоторых задачах будет играть роль скорость ввода, поэтому небольшой лайфхак, как сделать работать iostream быстрее: ios::sync_with_stdio(false); cin.tie(NULL);. Первая команда отключает синхронизацию ввода и вывода iostream (cin) и stdio (scanf), вторая отключает сброс буфера вывода при вводе. После этих команд ввод начинает работать быстрее, но после них нельзя использовать вперемешку cin и scanf, а также cout и printf.
- Когда-нибудь добавим автоматическую МЫ проверку на значит, что сейчас надо писать код как попало. Очень рекомендуется следонекоторым правилам. Мы все будем придерживаться Google C++(https://google.github.io/styleguide/cppguide.html). Это не будет проверяться вручную, а только автоматически, когда мы сможем включить эту опцию. Это может быть немного раздражительно следовать этим правилам (так же, как и правилам с жесткими ошибками компиляции из-за предупреждений), но в любой команде, где вы разрабатываете код не в одиночку, нужно уметь следовать стилю. Поэтому мы приняли один стиль, который все будут стараться соблюдать. Это не значит, что по-другому писать нельзя, но надо было выбрать и мы выбрали именно такой.

Задача А. Стек на векторе

Имя входного файла: stack1.in
Имя выходного файла: stack1.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

В этой задаче обязательно использовать самостоятельно написанный вектор.

Реализуйте работу стека. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "+ N", либо "-". Команда "+ N" означает добавление в стек числа N, по модулю не превышающего 10^9 . Команда "-" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека.

Формат входных данных

В первой строке входного файла содержится количество команд — M ($1 \le M \le 10^6$). Каждая последующая строка исходного файла содержит ровно одну команду.

Формат выходных данных

Выведите числа, которые удаляются из стека, по одному в каждой строке. Гарантируется, что изъятий из пустого стека не производится.

stack1.in	stack1.out
6	10
+ 1	1234
+ 10	
-	
+ 2	
+ 1234	
-	

Задача В. Стек на списке

Имя входного файла: stack2.in
Имя выходного файла: stack2.out
Ограничение по времени: 2 секунды
Ограничение по памяти: 64 мегабайта

В этой задаче обязательно использовать самостоятельно написанный односвязный список.

Реализуйте работу стека. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "+ N", либо "-". Команда "+ N" означает добавление в стек числа N, по модулю не превышающего 10^9 . Команда "-" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека.

Формат входных данных

В первой строке входного файла содержится количество команд — M ($1 \le M \le 10^6$). Каждая последующая строка исходного файла содержит ровно одну команду.

Формат выходных данных

Выведите числа, которые удаляются из стека, по одному в каждой строке. Гарантируется, что изъятий из пустого стека не производится.

stack2.in	stack2.out
6	10
+ 1	1234
+ 10	
-	
+ 2	
+ 1234	
-	

Задача С. Очередь на векторе

Имя входного файла: queue1.in Имя выходного файла: queue1.out Ограничение по времени: 2 секунды Ограничение по памяти: 64 мегабайта

В этой задаче обязательно использовать самостоятельно написанный вектор. Вы можете использовать любую стратегию увеличения/уменьшения размера массива, но размер массива должен быть не больше Cn, где n — количество элементов в очереди, для некоторого C>1.

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "+ N", либо "-". Команда "+ N" означает добавление в очередь числа N, по модулю не превышающего 10^9 . Команда "-" означает изъятие элемента из очереди.

Формат входных данных

В первой строке содержится количество команд — M ($1 \le M \le 10^6$). В последующих строках содержатся команды, по одной в каждой строке.

Формат выходных данных

Выведите числа, которые удаляются из очереди, по одному в каждой строке. Гарантируется, что извлечения из пустой очереди не производится.

queue1.in	queue1.out
4	1
+ 1	10
+ 10	
-	
-	

Задача D. Очередь на списке

Имя входного файла: queue2.in Имя выходного файла: queue2.out Ограничение по времени: 2 секунды Ограничение по памяти: 64 мегабайта

В этой задаче обязательно использовать самостоятельно написанный связный список.

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо "+ N", либо "-". Команда "+ N" означает добавление в очередь числа N, по модулю не превышающего 10^9 . Команда "-" означает изъятие элемента из очереди.

Формат входных данных

В первой строке содержится количество команд — M ($1 \le M \le 10^6$). В последующих строках содержатся команды, по одной в каждой строке.

Формат выходных данных

Выведите числа, которые удаляются из очереди, по одному в каждой строке. Гарантируется, что извлечения из пустой очереди не производится.

queue2.in	queue2.out
4	1
+ 1	10
+ 1 + 10	
-	
-	

Задача Е. Правильная скобочная последовательность

Имя входного файла: brackets.in Имя выходного файла: brackets.out Ограничение по времени: 2 секунды Ограничение по памяти: 64 мегабайта

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов '(', ')', '[' и ']'. Выясните, является ли она правильной скобочной последовательностью с двумя типами скобок.

Используйте реализацию стека из задачи 1 или 2, на ваше усмотрение.

brackets.in	brackets.out
()()	YES
([])	YES
([)]	NO
((]]	NO
)(NO

Задача F. Постфиксная запись

Имя входного файла: postfix.in Имя выходного файла: postfix.out Ограничение по времени: 1 секунда Ограничение по памяти: 64 мегабайта

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел A и B записывается как A B +. Запись B C + D * обозначает привычное нам (B+C)*D, а запись A B C + D * + означает A+(B+C)*D. Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

Формат входных данных

В единственной строке записано выражение в постфиксной записи, содержащее однозначные числа и операции +, -, *. Строка содержит не более 100 чисел и операций.

Формат выходных данных

Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений по модулю меньше 2^{31} .

postfix.in	postfix.out
8 9 + 1 7 - *	-102

Задача G. Минимум в очереди

Имя входного файла: queuemin2.in Имя выходного файла: queuemin2.out Ограничение по времени: 4 секунды Ограничение по памяти: 512 мегабайт

Вам дан массив $x[1\dots n]$ и число m. Для всех i от 1 до n-m+1 найдите миниум из чисел $x[i],x[i+1],\dots,x[i+m-1]$. Вам требуется вывести сумму всех таких минимумов.

Используете очередь с поддержкой операции "минимум в очереди".

Формат входных данных

Первая строка входного файла содержит три числа: n, m и k $(1 \le n \le 30\,000\,000, 1 \le m \le n, 2 \le k \le \min(n, 1000))$. Вторая строка содержит три целых числа: a, b и c $(-2^{31} \le a, b, c \le 2^{31} - 1)$. Третья строка содержит k целых чисел: $x[1], x[2], \ldots, x[k]$ $(-2^{31} \le x[i] \le 2^{31} - 1)$.

Остальные элементы массива вычисляются по следующей формуле: $x[i] = f(a \cdot x[i-2] + b \cdot x[i-1] + c)$. Здесь f(y) возвращает такое число $-2^{31} \leqslant z \leqslant 2^{31} - 1$, что y-z делится на 2^{32} .

Формат выходных данных

Выведите одно число — сумму минимумов на всех отрезках длины m.

queuemin2.in	queuemin2.out
10 3 2	33
1 1 0	
0 1	
1000000 15 5	-1879262596173354
283471207 23947205 3	
17625384 939393931 1838388 912740247 290470294	

Задача Н. Интерпретатор языка Quack

Имя входного файла: quack.in Имя выходного файла: quack.out Ограничение по времени: 2 секунды Ограничение по памяти: 64 мегабайта

Язык Quack — забавный язык, который фигурирует в задаче G с IPSC 2004. В этой задаче вам требуется написать интерпретатор языка Quack.

Виртуальная машина, на которой исполняется программа на языке Quack имеет внутри себя очередь, содержащую целые числа по модулю 65536 (get в описании операций означает извлечение из очереди, put — добавление в очередь). Кроме того, у виртуальной машины есть 26 регистров, которые обозначаются буквами от а до z.

+	Сложение: get x, get y, put (x+y) modulo 65536
-	Вычитание: get x, get y, put (x-y) modulo 65536.
*	Умножение: get x, get y, put (x*y) modulo 65536.
/	Целочисленное деление: get x, get y, put x div y. (будем
,	считать, что 0 div $0=0$)
%	Взятие по модулю: get x, get y, put x modulo y. (будем
/6	ражине по модулю, get x, get y, put x modulo y. (будем считать, что $0 \mod 0 = 0$)
>[register]	· ·
	Положить в регистр: get x, установить значение [register]
Z[:-t-]	B X.
<[register]	Взять из регистра: put значение [register]
P	Напечатать: get x, вывести x в стандартный поток вывода
	и перевести строку.
P[register]	Вывести значение регистра [register] в стандартный поток
	вывода и перевести строку.
C	Вывести как символ: get x, вывести символ с ASCII кодом
	х modulo 256 в стандартный поток вывода.
C[register]	Вывести регистр как символ: вывести символ с ASCII ко-
	дом х modulo 256 (где х — значение регистра [register]) в
	стандартный поток вывода.
:[label]	Метка: эта строка программы имеет метку [label].
J[label]	Переход на строку с меткой [label].
Z[register][label]	Переход если 0: если значение регистра [register] рав-
1. 0 11. 1	но нулю, выполнение программы продолжается с метки
	[label].
E[register1][register2][label]	Переход если равны: если значения регистров [register1]
	и [register2] равны, исполнение программы продолжается
	с метки [label].
G[register1][register2][label]	Переход если больше: если значение регистра [register1]
[[] [] [] [] []	больше, чем значение регистра [register2], исполнение
	программы продолжается с метки [label].
Q	Завершить работу программы. Работа также завершает-
	ся, если выполнение доходит до конца программы.
[number]	Просто число во входном файле — put это число.
	Tipoeto meno bo brodnom quinte put ito inento.

Формат входных данных

Входной файл содержит корректную синтаксически программу на языке Quack. Известно, что программа завершает работу не более чем за $100\,000$ шагов.

Формат выходных данных

Выведите содержимого стандартного потока вывода виртуальной машины в выходной файл.

Пример

quack.in	quack.out
100	5050
0	
:start	
>a	
Zaend	
<a>	
<a>	
1	
+	
-	
>b	
 b	
Jstart	
:end	
P	

Второй пример подразумевает UNIX-переводы строки (один символ с кодом 10).

quack.in quack.out 58 58 49 49 10 10 62 62 97 97 10 10 80 80 97 97 10 10 90 90 97 97 10 10 60 60 97 97 10 10 60 60 97 97 10 10 74 74 49 49 10 10 58 58 50 50 10 10 58 58 51 51 10 10 62 97 97 97 10 10 67 97 97 97 10 10 67 </th <th></th> <th>73 71 11 7 11</th>		73 71 11 7 11
88 49 49 10 62 97 10 10 80 80 97 97 10 10 90 90 97 97 50 50 10 10 60 97 10 10 60 97 10 10 74 49 49 49 10 10 58 58 50 50 10 10 48 48 10 10 58 58 51 10 62 97 97 97 10 10 90 90 97 97 52 52 10 10 67 67 97 97 52 52 10 10 67 67	quack.in	quack.out
49		
10	40	140
62 62 97 10 80 97 10 80 97 97 10 10 90 90 97 50 10 10 60 60 97 97 10 10 74 74 49 49 10 10 58 58 50 50 10 10 58 58 50 10 10 10 88 58 51 51 10 10 62 62 97 97 52 62 10 10 97 97 97 97 10 10 74 74 51 10 0 11 2a <t< td=""><td></td><td></td></t<>		
97 10 80 80 97 10 10 10 10 90 97 97 50 50 50 10 10 60 60 97 10 10 10 60 60 97 10 10 10 10 60 60 97 10 10 10 10 60 60 97 10 10 10 10 60 60 97 10 10 10 60 60 97 10 10 10 60 60 97 10 10 10 60 60 97 10 10 10 60 60 97 10 10 10 60 60 97 10 10 10 48 48 48 10 10 10 60 60 90 90 90 97 97 10 10 10 10 90 90 97 97 10 10 10 10 74 74 74 74 74 74 74 74 74 74 74 74 74	10	60
10	62	
80 80 97 97 10 10 90 97 97 97 10 10 10 10 10 10 10 1		
97		
10 90 97 97 50 10 10 10 10 60 60 97 97 10 10 10 74 49 49 10 10 58 58 58 50 10 10 10 62 97 97 10 10 58 58 51 11 10 10 62 97 97 97 10 10 10 62 97 97 10 10 10 58 58 58 51 10 10 10 58 58 51 10 10 10 58 58 58 51 10 10 10 10 58 58 58 58 51 10 10 10 10 58 58 58 58 51 10 10 10 10 90 90 97 97 10 10 10 90 90 90 97 97 10 10 10 10 74 74 51 51 10 10 74 74 51 51 10 10 58 58 58 58 58 58 58 52 10 0 0 0 11 12 2a	80	80
10 90 97 97 50 10 10 10 10 60 60 97 97 10 10 10 74 49 49 10 10 58 58 58 50 10 10 10 62 97 97 10 10 58 58 51 11 10 10 62 97 97 97 10 10 10 62 97 97 10 10 10 58 58 58 51 10 10 10 58 58 51 10 10 10 58 58 58 51 10 10 10 10 58 58 58 58 51 10 10 10 10 58 58 58 58 51 10 10 10 10 90 90 97 97 10 10 10 90 90 90 97 97 10 10 10 10 74 74 51 51 10 10 74 74 51 51 10 10 58 58 58 58 58 58 58 52 10 0 0 0 11 12 2a	97	97
90 97 50 10 10 10 60 60 97 97 10 10 10 74 49 49 10 10 58 50 50 10 48 48 48 10 10 58 58 58 58 58 58 58 58 58 59 10 10 48 49 10 10 58 62 62 97 97 10 10 10 62 62 97 97 10 10 10 62 67 97 10 10 10 68 68 68 51 10 10 69 97 97 97 10 10 10 10 67 97 10 10 10 68 67 97 10 10 10 10 68 68 58 58 58 58 51 11 10 10 10 10 10 10 10 10 10 10 10 10	10	10
97 97 50 10 60 60 97 97 10 10 74 49 10 10 58 58 50 10 48 48 10 10 48 48 10 10 62 62 97 10 10 10 62 62 97 97 52 52 10 10 67 97 97 97 52 52 10 10 74 74 51 10 67 97 97 97 10 10 74 74 51 10 68 88 52 52 10 10 74 51 51 51 10 0		
50	97	
10 60 60 97 10 10 10 74 44 74 49 10 10 10 58 88 58 50 60 10 10 10 48 10 10 10 62 62 62 97 97 10 10 10 90 90 97 97 10 10 10 67 97 97 10 10 10 67 74 51 10 10 10 10 67 74 51 11 10 10 10 10 10 58 88 58 52 10 10 10 10 67 97 97 10 10 10 10 67 97 97 10 10 10 10 10 67 97 97 10 10 10 10 10 67 97 97 10 10 10 10 10 10 10 10 10 10 10 10 10		
60 97 10 97 10 10 74 49 49 49 10 10 58 58 58 50 10 10 48 48 10 10 58 58 51 51 10 10 62 97 10 10 10 62 97 10 10 10 67 97 10 10 10 67 67 97 10 10 10 67 45 52 10 10 10 67 67 97 10 10 10 67 67 97 10 10 10 67 67 97 10 10 10 67 45 51 10 10 67 47 52 10 10 10 67 67 97 10 10 10 10 67 40 58 58 58 58 58 58 58 58 59 10 10 10 67 97 10 10 10 67 40 10 10 67 41 11 11 12 22 22 22 22 22 22 22 22 22 22	10	10
97 10 74 49 49 49 10 10 58 58 50 50 10 48 48 48 10 10 58 58 51 10 10 62 62 97 10 10 10 62 62 97 10 10 10 90 90 97 52 52 52 10 10 10 10 67 67 97 10 10 10 58 58 58 51 10 10 67 97 10 10 10 10 68 58 58 51 10 10 00 11 10 10 10 10 10 10 10 10 10		
10 74 49 49 10 10 58 58 50 10 48 48 10 10 58 58 51 10 62 97 10 10 62 97 97 90 90 90 97 97 52 52 10 10 67 97 97 97 10 10 67 97 97 97 10 10 68 58 52 52 10 10 74 51 10 10 74 51 51 51 10 10 68 58 52 52 10 10 74 51 51 51 10 10 68 58 52 52 10 10 6 62 97 97 97 97 97	60	60
74 49 10 10 58 58 50 10 48 48 10 10 58 58 51 51 10 10 62 62 97 10 90 90 97 97 52 52 10 10 67 97 97 97 10 10 67 97 10 10 74 74 51 51 10 10 58 58 52 52 10 10 74 74 51 51 10 0 10 10 0 0 11 11 2a 2a 2a2 2a 4a 2a 2a 2a 2a2 4a 3a 3a 2a4 2a 2a 2a 2a4 2a 2a4 2a4 2a 2a 2a4 </td <td>97</td> <td></td>	97	
49 10 10 58 58 50 50 10 48 48 48 10 10 58 58 58 58 58 58 58 51 11 10 10 62 62 97 10 10 10 90 90 97 97 10 10 10 67 97 10 10 10 67 97 10 10 10 68 58 58 58 58 58 52 10 0 0 11 10 10 58 58 58 58 58 58 58 58 58 58 58 58 52 10 10 10 10 10 10 10 10 10 10 10 10 10		
49 10 10 58 58 50 50 10 48 48 48 10 10 58 58 58 58 58 58 58 51 11 10 10 62 62 97 10 10 10 90 90 97 97 10 10 10 67 97 10 10 10 67 97 10 10 10 68 58 58 58 58 58 52 10 0 0 11 10 10 58 58 58 58 58 58 58 58 58 58 58 58 52 10 10 10 10 10 10 10 10 10 10 10 10 10		
10		
58 58 50 10 10 10 48 48 10 10 58 58 51 11 10 62 97 97 10 10 90 90 97 97 52 10 10 10 67 67 97 10 74 74 51 10 10 10 58 58 52 10 0 0 11 10 58 58 52 52 10 10 0 0 11 2a 2a2 2a 4a 31 31 31 31 31 32 3a 2a4 2a4 Ca Ca 33 33	10	
50 10 48 48 10 10 58 58 51 10 62 97 10 10 90 90 97 97 52 52 10 10 67 97 10 10 67 97 10 10 74 51 51 10 58 58 52 52 10 10 0 10 0 11 >a >a Pa Za2 <a< td=""> <a< td=""> J1 J1 12 12 0 0 13 >a 2a4 Ca4 Ca Ca4 J3 J3</a<></a<>		
10 48 10 48 10 10 58 58 58 51 10 10 62 62 97 97 10 10 90 90 97 52 10 10 67 97 10 10 67 97 10 10 10 68 58 58 52 10 10 10 58 58 58 52 10 10 10 58 58 58 52 10 10 10 58 58 58 52 10 10 10 58 58 52 10 10 10 58 58 52 10 10 10 58 58 52 10 10 10 58 58 52 10 10 10 10 58 58 52 10 10 10 10 58 58 52 10 10 10 10 10 58 58 52 10 10 10 10 10 10 10 10 10 10 10 10 10	50	
48 48 10 58 51 51 10 10 62 62 97 10 90 90 97 97 52 52 10 10 67 97 10 10 74 74 51 10 58 58 52 52 10 10 58 58 52 52 10 0 :1 :1 >a >a Pa Za2 <a< td=""> <a< td=""> JI :1 :2 0 :3 >a Za4 Za4 Ca J3</a<></a<>	10	
10 58 51 51 10 10 62 62 97 97 10 10 90 90 97 52 10 10 67 97 10 67 97 10 74 74 51 10 58 58 52 52 10 10 0 11 58 58 52 52 10 0 11 10 0 11 >a >a Pa Pa Za2 <a< td=""> 4 J1 11 11 12 12 0 0 13 >a 2a4 <a< td=""> 2a4 <a< td=""> 2a4 <a< td=""> 2a4 <a< td=""> 33 33</a<></a<></a<></a<></a<>		
58 58 51 10 62 62 97 97 10 90 90 90 97 97 52 52 10 67 97 10 74 74 51 10 58 58 52 52 10 10 58 58 52 52 10 10 0 11 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 0 0 :3 >a >a Za4 Ca Ca J3</a<></a<>		
51 10 62 62 97 97 10 90 90 90 97 97 52 10 67 67 97 10 74 74 51 10 74 51 10 10 58 58 52 10 10 0 11 >a >a Pa Za2 <a< td=""> 4a J1 1:2 .2 0 .3 >a .3 >a .2a Za4 .3 Ca .33 >a .2a J3 .3 J3 .3</a<>	10	
10 62 97 97 10 90 90 90 97 52 10 10 67 97 10 10 74 74 51 10 10 10 58 58 52 52 10 10 58 52 10 0 0 11 >a >a Pa Za2 <a< td=""> <a< td=""> J1 .2 0 .3 >a .3 >a .3 >a .2a4 Ca .0 .3 .a Za4 .a Za5 .a</a<></a<>	58	
62 97 10 90 90 97 52 10 67 97 52 10 67 97 10 10 74 51 10 58 51 10 58 52 10 0 0 :1 >a Pa Za2 <a 0="" :2="" :3="" <a="" ji="" sa2="">a Za4 Ca Ji Ji	51	51
62 97 10 90 90 97 52 10 67 97 52 10 67 97 10 10 74 51 10 58 51 10 58 52 10 0 0 :1 >a Pa Za2 <a 0="" :2="" :3="" <a="" ji="" sa2="">a Za4 Ca Ji Ji	10	
97 10 90 90 97 52 10 67 97 52 10 67 97 10 74 51 10 58 51 10 58 52 10 0 10 58 52 10 10 58 52 10 10 58 52 10 10 58 52 10 10 58 52 10 10 0 11 58 52 10 10 0 11 58 52 10 10 0 11 10 10 58 58 52 10 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 0 11 10 11 11	62	62
10 90 97 90 52 52 10 67 97 97 10 74 51 10 74 51 58 58 52 10 10 0 :1 :1 >a Pa Za2 <a< td=""> <a< td=""> J1 :2 0 :3 >a Za4 Ca Ca J3</a<></a<>	97	97
90 90 97 97 52 10 67 97 10 67 97 10 74 51 51 10 58 58 52 10 0 10 0 11 >a Pa Za2 <a< td=""> 4a J1 11 11 2a Za2 <a< td=""> J1 11 11 12 2 0 0 13 >a Za4 Ca Ca J3</a<></a<>	10	
97		
52 52 10 67 67 97 10 97 10 10 74 51 51 10 58 58 52 52 10 0 11 10 0 10 0 11 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 .2 0 .3 >a .3 >a .a Za4 .a Ca .a J3 J3</a<></a<>	90	90
10 67 67 97 97 10 74 51 10 58 58 52 52 10 0 0 10 0 0 :1 >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 0 :3 >a Za4 <a< td=""> Ca <a< td=""> J3 J3</a<></a<></a<></a<>	97	
67 97 10 74 10 74 51 10 58 58 58 52 10 0 0 :1 >a Pa Za2 <a 0="" :2="" :3="" j1="">a J1 :2 0 0 :3 >a Za4 Ca J3 J3 J3	52	
97 10 74 74 51 51 10 58 52 52 10 10 0 10 0 10 0 11 >a >a Pa Pa Za2 Za2 <a< td=""> <a< td=""> J1 J1 :2 2 0 0 :3 >a Za4 Za4 Ca J3</a<></a<>	10	
10 74 51 51 10 10 58 58 52 52 10 0 0 :1 >a >a Pa Za2 <a< td=""> 2a2 <a< td=""> 3 J1 1 :2 2 0 3 >a 2a4 Ca Ca J3 J3</a<></a<>	67	67
10 74 51 51 10 10 58 58 52 52 10 0 0 :1 >a >a Pa Za2 <a< td=""> 2a2 <a< td=""> 3 J1 1 :2 2 0 3 >a 2a4 Ca Ca J3 J3</a<></a<>		97
74 51 51 10 10 58 58 52 52 10 10 0 0 :1 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 0 :3 >a Za4 Za4 Ca J3 J3 J3</a<></a<>	10	10
51 10 58 58 52 52 10 10 0 0 :1 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 0 0 :3 >a >a Za4 Ca Ca J3</a<></a<>	74	74
10 10 58 58 52 52 10 10 0 0 :1 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 0 :3 :3 >a >a Za4 Ca Ca J3</a<></a<>	51	51
58 52 52 10 10 0 11 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 0 :3 >a Za4 Za4 Ca J3 J3 J3</a<></a<>	10	10
52 52 10 0 0 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 :3 >a >a Za4 Ca Ca J3 J3 J3</a<></a<>		
10 10 0 0 :1 :1 >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 :3 >a >a Za4 Ca Ca J3</a<></a<>		
0 :1 >a >a Pa Pa Za2 Za2 <a< td=""> <a< td=""> J1 J1 :2 :2 0 :3 >a >a Za4 Ca Ca J3</a<></a<>	52	
:1 >a >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 0 :3 >a >a Za4 Ca Ca J3 J3</a<></a<>		
:1 >a >a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 0 :3 >a >a Za4 Ca Ca J3 J3</a<></a<>	0	
>a >a Pa Pa Za2 <a< td=""> <a< td=""> J1 :2 :2 0 0 :3 >a >a >a Za4 Ca Ca J3</a<></a<>	:1	
Pa Za2 <a< td=""> J1 :2 :2 0 0 :3 :3 >a Za4 Ca J3 J3 J3</a<>		
Za2 <a< td=""> J1 J1 :2 :2 0 0 :3 :3 >a >a Za4 Za4 Ca Ca J3 J3</a<>		
<a< td=""> <a< td=""> J1 :2 0 :2 0 0 :3 :3 >a >a Za4 Za4 Ca Ca J3 J3</a<></a<>		
J1 :2 0 :2 0 :3 >a :3 Za4 Za4 Ca Ca J3 J3		
:2 :2 0 0 :3 :3 >a >a Za4 Za4 Ca Ca J3 J3		
0 :3 :3 :3 >a >a Za4	J1	
:3 :3 >a >a Za4 Za4 Ca Ca J3 J3		
:3 >a Za4 Za4 Ca Ca J3 J3	0	0
>a >a Za4 Za4 Ca Ca J3 J3	:3	
Za4 Za4 Ca Ca J3 J3		
Ca Ca J3 J3		
J3 J3		
.4		
	÷#	1 th