

Report on
The 2000 ACM Southern Africa Programming Contest
Sponsored by IBM
Hosted by the University of Pretoria
Additional locations at Rhodes University and the University of Cape Town
Date of Contest: Saturday, 16 September 2000

On Saturday the 16th of September, students from all over Southern Africa participated simultaneously in three different locations: Pretoria, Grahamstown and Cape Town. A total of 50 teams representing 5 different universities participated in the contest.

Final Standings

Unfortunately, due to a power failure at the Cape Town site, the contest was terminated (at all three sites) one hour early. The standings at that time are taken as the final standings. The following standings are final (in terms of teams), even though some team members names have not been entered.

Position	Team name	Team members	University	Problems solved	Total time (incl. penalties)
1	WITS 38	Paul Cook, Greg Kempe, Hein Kruger	Witwatersrand	4	5h41
2	UCT 11	Carl Hultquist, David Turner, Rainer Hoft	Cape Town	4	8h20
3	TUKS 6	Gerhard Bijker, Danie Conradie, Jaco Cronje	Pretoria	3	4h10
4	UCT 9	Cobus Combrink, Tim Pauw, Carl Scheffler	Stellenbosch	3	7h18
5	C-Sick	Jan Gutter, Johan van der Berg, Leon Wabeke	Pretoria	3	7h45
6	Burning Chrome	Dylan Shell, George Christelis, George Konidaris	Witwatersrand	2	1h59

The problems

1. [Wrap it up.](#)
2. [Compiler code generation.](#)
3. [Crosswords.](#)
4. [Genetic family tree.](#)
5. [Fire trucks.](#)
6. [Message decoding.](#)

A word of thanks

Nanette Saes functioned as the very capable day manager, not only throughout the day itself, but in all the preparations leading up to it.

Linda Marshall, Nicolet Ros, Frans van den Bergh and Gerrie Swart provided assistance in fending off tough questions, taking in submissions and generally keeping track of everything.

Technical support was provided by Chris Potgieter and his team who were there to smooth out any hardware and software glitches.

As for food and entertainment, Manhattan Bagels of Hatfield Plaza made sure that the energy level of the contestants stayed high with a timely supply of fresh bagels and muffins, Hotel 224 served up a great braai to fill in the gaps.

Of course all of this would not have been possible without the sponsors: IBM provided the main sponsorship for the contest, with local support provided by WebPort.

Second Southern African Regional ACM Collegiate Programming Competition

Sponsored by IBM

www.ibm.com

National support from WebPort

www.webport.co.za

Problem 1/Red Balloon: Wrap it up

Saturday 16 September 2000

You've been given the unfortunate task of planning text-book wrapping for your brothers and sisters (still in high-school). To save money, your parents bought sufficient plastic covering for any conceivable number of text-books (certainly enough for all of the wrapping you will be asked to do).

The plastic comes in a huge roll (which is guaranteed to be long enough) 1,8m wide. Your parents have also insisted that you plan the book covering so that a stretch of the roll will be cut off *straight* (meaning that the cut line will be at 90 degrees to the edges). To save time, you will be wrapping each book with a rectangular piece of plastic — in particular, you will not be making any further cutouts to beautify the wrapping.

Given the n text-books to be covered, your task is to determine the minimum length k (in millimeters) of plastic to cut from the roll, so that each book is covered with a 4cm wrap-around (perhaps with some wastage). Each of the n books has the usual shape for books (rectangular, and having both front and back covers) and is described using an ordered sequence of three integers (measured in millimeters): the height, width, and thickness. All except the thickness are guaranteed to be positive. (The thickness may, in fact, be zero in the case that you are asked to cover a leaflet.) You are guaranteed never to have a book with a dimension greater than 1720mm.

Your input will consist of a sequence of lines: the first line contains n ; each of the n subsequent lines (terminated by a new-line) contains three ordered integers (separated by a space) describing the height, width and thickness of one of the books to be covered.

Your output will consist of integer k followed by a new-line.

Sample input:

```
2
620 500 0
610 300 400
```

Corresponding output:

```
1080
```

**Second Southern African Regional
ACM Collegiate Programming
Competition**

Sponsored by IBM

`www.ibm.com`

National support from WebPort

`www.webport.co.za`

Problem 2/Yellow Balloon: Compiler code generation

Saturday 16 September 2000

You have been asked to create a code generator for a simple machine. The input to the code generator (produced by the compiler's front-end) consists of arithmetic expressions in postfix form (also known as *post-order* or *reverse Polish notation* form). The output will be in the assembly language of the target machine.

The target machine has a single register and a number of temporary locations (which are allocated by the *assembler* — something beyond the scope of this problem) named \$1, . . . , \$9 (to be used in that order). The machine supports the following instructions, where the operand is either an identifier (identifiers consist of single letters — either upper and lower case) or a temporary location:

- L — load the operand into the register.
- A — add the operand to the contents of the register.
- S — subtract the operand from the contents of the register.
- M — multiply the contents of the register by the operand.
- D — divide the contents of the register by the operand.
- R — give the remainder of the contents of the register divided by the operand.
- ST — store the contents of the register in the operand location.

The arithmetic operators replace the contents of the register with the expression result.

The input consists of a single, well-formed (and non-empty), postfix expression appearing on one line with no white-space (no spaces, tabs, new-line

characters, etc.). The operators include the normal ones +, -, *, / and % (for remainder).

The output will consist of a sequence of lines each with a single instruction, satisfying the following:

1. There will be a single space separating the instruction and the operand (if there is one).
2. Each line (including the last one) will be terminated by a new-line.
3. The original order of the operands must be preserved in the assembly code.
4. Assembly code must be generated for each operator as soon as it is encountered.
5. As few temporaries as possible should be used (within the limits of the above restrictions).
6. For each operator in the input expression, the minimum number of instructions must be generated (also within the limits of the above restrictions).
7. The final result must be left in the register.

Your code generator must create straightforward code, without any optimizations whatsoever.

Sample input:

AB+CD+EF++GH+++

Corresponding output:

```
L A
A B
ST $1
L C
A D
ST $2
L E
A F
A $2
ST $2
L G
A H
A $2
A $1
```

Second Southern African Regional ACM Collegiate Programming Competition

Sponsored by IBM

`www.ibm.com`

National support from WebPort

`www.webport.co.za`

Problem 3/Orange Balloon: Crosswords

Saturday 16 September 2000

In an effort to entertain your crossword-obsessed aunt and uncle, you decide to work on some software to create compact crosswords. A crossword must be rectangular and adjacent letters **must** be part of a word. the meaning of ‘adjacent’ differs for the two different types of crosswords — see below. This problem is case-sensitive.

There are two types of crosswords:

- strict — these allow words to be vertical and horizontal in the grid only; they may be top-to-bottom or vice-versa and left-to-right or right-to-left.
- diagonal — these **also** allow words to appear on a diagonal.

You are given a sequence of n words along with possible synonyms (meaning-equivalents) for that word — no word will appear more than once anywhere in the input. With this, you are to determine the minimum number k of black squares which *must* be present in a crossword for the words. Each word, or a synonym thereof, must appear exactly once in the crossword and no additional words may appear.

The input consists of a number of lines. The first line contains either the letter S or the letter D (designating whether you are to consider strict or

diagonal crosswords). The next line contains positive integer n . The next n lines (each terminated by a new-line) each consist of at least one word, followed optionally by synonym words (for the first word on the line), each separated by a space. The output consists of k followed by a newline.

Sample input:

```
S
Helicopter copter
election poll
```

Corresponding output:

```
12
```

since 'copter' and 'poll' can be intersected at either the 'o' or the 'p', in both cases giving a 5×4 crossword with $4 \cdot 5 - 5 - 4 + 1 = 12$.

Second Southern African Regional ACM Collegiate Programming Competition

Sponsored by IBM

`www.ibm.com`

National support from WebPort

`www.webport.co.za`

Problem 4/Blue Balloon: Genetic family tree

Saturday 16 September 2000

Genomic computing has become big business, and you decide to cash in. Your first attempt is to write a program which constructs a family tree from genetic information.

The gene sequence for an individual is given as a string (with no spaces) consisting of only the letters a, c, g, and t. The relatedness of two individuals can be estimated using the Hamming distance between their gene sequences. The Hamming distance between strings x and y is the number of individual letters in x which must be changed to give y . For Hamming distance purposes, you are guaranteed that x and y are the same length.

You may assume that each individual has either zero or two offspring, and that this species functions asexually (each individual has at most one parent). We are interested in a family tree in which we minimize the Hamming distance between a parent and its children.

Given $n : n > 0$ individual gene sequences, you are to determine a *plausible* family tree which has the smallest total Hamming distance h (the sum of the Hamming distances between all parents and their children). You are guaranteed that there is only one such tree. Your output must include the family tree.

The input consists of several lines. The first line contains n . Each of the following n lines contains a single individual's gene sequence. The output

consists of $n + 1$ lines. The first line contains h . The remaining lines contain the family tree in prefix form. Each line contains a single individual, followed by a space and either the letter P (if the individual is a parent) or the letter N (if the individual is not a parent). When there are two children, they must be output in lexicographic ('telephone book') order.

Sample input:

```
3
gattaca
gcttaca
gaccata
```

Corresponding output:

```
4
gattaca P
gaccata N
gcttaca N
```

**Second Southern African Regional
ACM Collegiate Programming
Competition**

Sponsored by IBM

`www.ibm.com`

National support from WebPort

`www.webport.co.za`

Problem 5/Green Balloon: Fire trucks

Saturday 16 September 2000

Your city's fire department collaborates with the transportation department to maintain maps of the city which reflects the current status (on that day) of the city streets. On any given day, some streets are closed for repairs or construction. The fire engine navigator needs to be able to select routes, from the firestation to the fire, that do not use closed streets. You must write a program that the central dispatcher can use to generate routes from the firestation to a fire.

Street intersections are identified by positive integers not more than 20, with the firestation always at intersection 1. The input file consists of a test case representing a fire. The first line consists of a single integer which is the number of the street intersection closest to the fire. The following lines each consist of a pair of positive integers separated by a space and terminated by a new-line. They represent the adjacent street intersections of an open street. For example, if 4 7 appears on a line, then the street between intersections 4 and 7 is open and there are no other intersections between 4 and 7 on that section of street. The final line consists of a pair of 0s.

Your output must consist of a line for each valid route from the firestation to the intersection closest to the fire, written with the intersections in the order in which they appear on the route. The intersections must appear separated by a space, terminated by a new-line. Include only routes which

do not pass through any streetintersection more than once. The lines of output must be in lexicographical order.

Sample input:

```
6
1 2
1 3
3 4
3 5
4 6
5 6
2 3
2 4
0 0
```

Sample output:

```
1 2 3 4 6
1 2 3 5 6
1 2 4 3 5 6
1 2 4 6
1 3 2 4 6
1 3 4 6
1 3 5 6
```

Second Southern African Regional ACM Collegiate Programming Competition

Sponsored by IBM

`www.ibm.com`

National support from WebPort

`www.webport.co.za`

Problem 6/Pink Balloon: Message decoding

Saturday 16 September 2000

You are to implement a message decoder for a scheme which sends messages in two parts: a header and the encoded message itself.

The foundation of the scheme is a sequence of binary ‘key’ strings as follows:

0, 00, 01, 10, 000, 001, 010, 011, 100, 101, 110, 0000, . . .

The first one is of length 1, the next three of length 2, the next seven of length 3, next fifteen of length 4, etc. If two adjacent keys have the same length, the second can be obtained from the first by adding 1. Note that there are no keys consisting only of 1s.

The keys are mapped to the characters in the header in order. Suppose the header is **Bruce**, the first key (0) is mapped to B, etc.

The encoded message contains only 0s and 1s and possibly carriage returns (new-lines) — though they are to be ignored. The message is divided into segments. The first 3 digits of a segment give the binary representation of the length of the keys in that segment. For example, if the first 3 digits are 011_2 , then the remainder of the segment consists of keys of length 3_{10} (one of 000, 001, . . . , 110). The end of a segment is a string of 1s which is the same length as the length of the keys in that segment (this is why all 1s were not used above). So, in the example above, the segment would be

terminated by 111. The entire message is terminated by a segment beginning 000 (which would signify a segment in which keys have length 0). The message is decoded by translating the keys in the segments into the header characters to which they have been mapped.

The input file contains several lines. The first line contains the header, terminated by a new-line (which is not part of the message); any spaces *are* part of the header. The length of the header is limited by the fact that key strings have a maximum length of 111_2 (7_{10} in decimal). If there are multiple copies of a character in the header, then several keys will map to that character.

The remaining lines of input contain the encoded message. The message is made up only of 0s and 1s (and possibly carriage-returns — to be ignored). You are guaranteed that it is a valid message according to the encoding described above.

Your output must consist of the message, followed by a new-line.

Sample input:

```
$##*/  
01  
00000  
1011011  
000111  
00101000
```

Corresponding output:

```
##*/$
```