Programming Challenges

Strings

Strings Representation

- Character Codes (ASCII)

0	NUL	1	SOH	2	STX	3	ETX	4	EOT	5	ENQ	6	ACK	7	BEL
8	BS	9	HT	10	NL	11	VT	12	NP	13	CR	14	SO	15	SI
16	DLE	17	DC1	18	DC2	19	DC3	20	DC4	21	NAK	22	SYN	23	ETB
24	CAN	25	EM	26	SUB	27	ESC	28	FS	29	GS	30	RS	31	US
32	SP	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	/	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	—	125	}	126	~	127	DEL

Strings Representation

- Characters properties:
  - All non-printable characters have either the first three bits as zero or all seven lowest bits as one.
  - Both the upper- and lowercase letters and the numerical digits appear sequentially.
  - Converting a character (say, “I”) to its rank.

Strings Representation

- Characters properties:
  - Converting character (say “C”) from upper- to lowercase.
  - The character code tells us what will happen when naively sorting text files.
  - Non-printable character codes for new-line (10) and carriage return (13).

Strings Representation

- ASCII and Unicode:
  - C/C++, Pascal
  - JAVA

Strings Representation

- Which String Representation?
  - Which uses the least amount of space?
  - Which constrains the content of the strings which can possibly be represented?
  - Which allow constant-time access to the ith character?
  - Which allow efficient checks for out-of-bounds errors?
  - Which allow efficient deletion or insertion?

Strings Operations

- Searching for Patterns:
  - Brute Force ?
  - Using algorithms (Boyer–Moore–Horspool , KMP) ?
  - What is the complexity ?
  - Should we use them ?

C Characters

- #include <cctype> /* include the character library */
- int isalpha(int c); /* true if c is either upper or lower case */
- int isupper(int c); /* true if c is upper case */
- int islower(int c); /* true if c is lower case */
- int isdigit(int c); /* true if c is a numerical digit (0-9) */
- int ispunct(int c); /* true if c is a punctuation symbol */
- int isxdigit(int c); /* true if c is a hexadecimal digit (0-9,A-F) */
- int isprint(int c); /* true if c is any printable character */
- int toupper(int c);
- int tolower(int c);

C Strings

- #include <cstring> /* include the string library */
- char *strcat(char *dst, const char *src);
- int strcmp(const char *s1, const char *s2);
- char *strcpy(char *dst, const char *src);
- size_t strlen(const char *s);
- char *strstr(const char *s1, const char *s2); /* search for s2 in s1 */
- char *strtok(char *s1, const char *s2); /* iterate words in s1 */

C++ Strings

- #include <string>
- string::size()
- string::empty() /* is it empty */
- string::c_str()
- string::operator [](size_type i) /* access the ith character */
- string::append(s) /* append to string */
- string::erase(n,m) /* delete a run of characters */

C++ Strings

- #include <string>
- string::insert(size_type n, const string&s) /* insert string s at n */
- string::find(s)
- string::rfind(s) /* search left or right for the given string */
- string::first()
- string::last() /* get characters, also there are iterators */

WERTYU

PC/UVa IDs: 110301/10082, **Popularity:** A,
Success rate: high **Level:** 1

WERTYU

A common typing error is to place your hands on the keyboard one row to the right of the correct position. Then “Q” is typed as “W” and “J” is typed as “K” and so on.

Your task is to decode a message typed in this manner.

Input

Input consists of several lines of text. Each line may contain digits, spaces, uppercase letters (except “Q”, “A”, “Z”), or punctuation shown above [except back-quote (‘)]. Keys labeled with words [Tab, BackSp, Control, etc.] are not represented in the input.

WERTYU

- *Output*
- You are to replace each letter or punctuation symbol by the one immediately to its left on the QWERTY keyboard shown above. Spaces in the input should be echoed in the output.

- *Sample Input*

  O S, GOMR YPFSU/
- *Sample Output*

  I AM FINE TODAY.

Sorting

- Sorting Applications?
  - *Uniqueness Testing*
  - *Deleting Duplicates*
  - *Median/Selection*
  - *Contest Scoreboard* ☺

Sorting

- Basic Sorting Algorithms:
  - *Selection Sort*
  - *Heap Sort*
  - *Bubble Sort*
  - *Insertion Sort*
  - *Quick Sort*

Sorting

- Multi-criteria Sorting:
  - How can we break ties in sorting using multiple criteria?
  - Using complex compare functions
  - Stable Sorting

Sorting in C/C++

- cstdlib:
  - bsearch:
    - void* bsearch (const void* key, const void* base, size_t num, size_t size, int (*compar)(const void*,const void*));
  - qsort:
    - void qsort(void *base, size_t nel, size_t width,int (*compare) (const void *, const void *));

Sorting in C/C++

- algorithm:
  - void sort (RandomAccessIterator first, RandomAccessIterator last);
  - void stable_sort (RandomAccessIterator first, RandomAccessIterator last);
  - void nth_element (RandomAccessIterator first, RandomAccessIterator nth, RandomAccessIterator last);
  - bool is_sorted (ForwardIterator first, ForwardIterator last);
  - unique (ForwardIterator first, ForwardIterator last);
  - bool binary_search (ForwardIterator first, ForwardIterator last, const T& val);

Vito's Family

PC/UVa IDs: 110401/10041, **Popularity:** A, **Success rate:** high **Level:** 1

The famous gangster Vito Deadstone is moving to New York. He has a very big family there, all of them living on Lamafia Avenue. Since he will visit all his relatives very often, he wants to find a house close to them.

Indeed, Vito wants to minimize the total distance to all of his relatives and has

blackmailed you to write a program that solves his problem.

Vito's Family

- *Input*

The input consists of several test cases. The first line contains the number of test cases. For each test case you will be given the integer number of relatives r ($0 < r < 500$) and the street numbers (also integers) $s1, s2, \ldots, si, \ldots, sr$ where they live ($0 < si < 30,000$).

Note that several relatives might live at the same street number.

- *Output*

For each test case, your program must write the minimal sum of distances from the optimal Vito's house to each one of his relatives. The distance between two street numbers si and sj is $dij = |si - sj|$.

Vito's Family

- *Sample Input*

2

2 2 4

3 2 4 6

- *Sample Output*

2

4