WTC Exam Questions

Level 00

1. only_z
   Write a program that displays an ‘z’’ character on the standard output.
2. only_a
   Write a program that displays an 'a' character on the standard output.
3. aff_z
   Write a program that takes a string, and displays the first 'z' character it encounters in it, followed by a newline. If there are no 'z' characters in the string, the program writes 'z' followed by a newline. If the number of parameters is not 1, the program displays 'z' followed by a newline.

   Example:

   $> ./aff_z "abc" | cat -e
   z$
   $> ./aff_z "dubO a POIL" | cat -e
   z$
   $> ./aff_z "zaz sent le poney" | cat -e
   z$
   $> ./aff_z | cat -e
   z$

4. hello
   Write a program that displays "Hello World!" followed by a \n.
5. maff_alpha
   Write a program that displays the alphabet, with even letters in uppercase, and
   odd letters in lowercase, followed by a newline.

   Example:

   $> ./maff_alpha | cat -e
   aBcDeFgHiJkLmNoPqRsTuVwXyZ$

6. maff_revalpha

Write a program that displays the alphabet in reverse, with even letters in uppercase, and odd letters in lowercase, followed by a newline.

Example:

$> ./maff_revalpha | cat -e
zYxWvUtSrQpOnMlKjIhGfEdCbA$

7. ft_countdown

Write a program that displays all digits in descending order, followed by a newline.

Example:
$> ./ft_countdown | cat -e
9876543210$
$>

8. ft_print_numbers

Write a function that displays all digits in ascending order.

Your function must be declared as follows:

```
void	ft_print_numbers(void);
```

Level 01

9. ft_putstr

Write a function that displays a string on the standard output.
The pointer passed to the function contains the address of the string's first character.
Your function must be declared as follows:

```
void	ft_putstr(char *str);
```

10. ft_strcpy

Reproduce the behavior of the function strcpy (man strcpy).

Your function must be declared as follows:

```
char	*ft_strcpy(char *s1, char *s2);
```

11. ft_swap

    Write a function that swaps the contents of two integers the adresses of which are passed as parameters.

    Your function must be declared as follows:

    ```
    void	ft_swap(int *a, int *b);
    ```

12. rev_print [No solution yet]

    Write a program that takes a string, and displays the string in reverse followed by a newline.

    If the number of parameters is not 1, the program displays a newline.

    Examples:

    ```
    $> ./rev_print "zaz" | cat -e
    zaz$
    $> ./rev_print "dub0 a POIL" | cat -e
    LIOP a 0bud$
    $> ./rev_print | cat -e
    $
    ```

13. rot_13

    Write a program that takes a string and displays it, replacing each of its letters by the letter 13 spaces ahead in alphabetical order.

    'z' becomes 'm' and 'Z' becomes 'M'. Case remains unaffected.

    The output will be followed by a newline.

    If the number of arguments is not 1, the program displays a newline.

    Example:

    ```
    $>./rot_13 "abc"
    Nop
    $>./rot_13 "My horse is Amazing." | cat -e
    Zl ubefr vf Nznmvat.$
    $>./rot_13 "AkjhZ zLKIJz , 23y " | cat -e
    NxwuM mYXVWm , 23l $
    ```

$>./rot_13 | cat -e
$
$>
$>./rot_13 "" | cat -e
$
$>

14. ulstr

Write a program that takes a string and reverses the case of all its letters. Other characters remain unchanged.

You must display the result followed by a '\n'.

If the number of arguments is not 1, the program displays '\n'.

Examples :

$>./ulstr "L'eSPrit nE peUt plUs pRogResSer s'Il staGne et sI peRsIsTent VAnIte et auto-justification." | cat -e
l'EspRIT Ne PEuT PLuS PrOGrESsER S'iL STAgNE ET Si PErSiStENT vaNiTE ET AUTO-JUSTIFICATION.$
$>./ulstr "S'enTOuRer dE sECreT eSt uN sIGnE De mAnQuE De coNNaiSSanCe. " | cat -e
s'ENtoUrER De SecREt EsT Un SigNe dE MaNqUe dE COnnAIssANcE. $
$>./ulstr "3:21 Ba tOut moUn ki Ka di KE m'en Ka fe fot" | cat -e
3:21 bA ToUT MOuN KI kA DI ke M'EN kA FE FOT$
$>./ulstr | cat -e
$

15. Search_and_replace [No solution yet]

Write a program called search_and_replace that takes 3 arguments, the first arguments is a string in which to replace a letter (2nd argument) by another one (3rd argument).

If the number of arguments is not 3, just display a newline.

If the second argument is not contained in the first one (the string) then the program simply rewrites the string followed by a newline.

Examples:
$>./search_and_replace "Papache est un sabre" "a" "o"
Popoche est un sobre
$>./search_and_replace "zaz" "art" "zul" | cat -e
$
$>./search_and_replace "zaz" "r" "u" | cat -e

```
zaz$
$>./search_and_replace "jacob" "a" "b" "c" "e" | cat -e
$
$>./search_and_replace "ZoZ eT Dovid oiME le METol." "o" "a" | cat -e
ZaZ eT David aiME le METal.$
$>./search_and_replace "wNcOre Un ExEmPle Pas Facilw a Ecrirw " "w" "e" | cat -e
eNcOre Un ExEmPle Pas Facile a Ecrire $
```

16. first_word

Write a program that takes two strings and displays, without doubles, the characters that appear in either one of the strings.

The display will be in the order characters appear in the command line, and will be followed by a \n.

If the number of arguments is not 2, the program displays \n.

Example:

```
$>./union zpadinton "paqefwtdjetyiytjneytjoeyjnejeyj" | cat -e
zpadintoqefwjy$
$>./union ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
df6vewg4thras$
$>./union "rien" "cette phrase ne cache rien" | cat -e
rienct phas$
$>./union | cat -e
$
$>
$>./union "rien" | cat -e
$
$>
```

17. rotone

Write a program that takes a string and displays it, replacing each of its letters by the next one in alphabetical order.

'z' becomes 'a' and 'Z' becomes 'A'. Case remains unaffected.

The output will be followed by a \n.

If the number of arguments is not 1, the program displays \n.

Example:

```
$>./rotone "abc"
bcd
$>./rotone "Les stagiaires du staff ne sentent pas toujours tres bon." | cat -e
Mft tubhjbjsft ev tubgg of tfoufou qbt upvkpvst usft cpo.$
$>./rotone "AkjhZ zLKIJz , 23y " | cat -e
BlkiA aMLJKa , 23z $
$>./rotone | cat -e
$
$>
$>./rotone "" | cat -e
$
$>
```

Level 02

18. ft_strcmp

    Reproduce the behavior of the function strcmp (man strcmp).

    Your function must be declared as follows:

    ```
    int    ft_strcmp(char *s1, char *s2);
    ```

19. ft_atoi

    Write a function that converts the string argument str to an integer (type int) and returns it.

    It works much like the standard atoi(const char *str) function, see the man.

    Your function must be declared as follows:

    ```
    int         ft_atoi(const char *str);
    ```

20. aff_first_param

    Write a program that takes strings as arguments, and displays its first argument followed by a \n.

    If the number of arguments is less than 1, the program displays \n.

Example:

```
$> ./aff_first_param vincent mit "l'ane" dans un pre et "s'en" vint | cat -e
vincent$
$> ./aff_first_param "j'aime le fromage de chevre" | cat -e
j'aime le fromage de chevre$
$> ./aff_first_param
$
```

21. print_bits

Write a function that takes a byte, and prints it in binary WITHOUT A NEWLINE AT THE END.

Your function must be declared as follows:

```
void	print_bits(unsigned char octet);
```

Example, if you pass 2 to print_bits, it will print "00000010"

22. ft_strrev

Write a function that reverses (in-place) a string.

It must return its parameter.

Your function must be declared as follows:

```
char	*ft_strrev(char *str);
```

23. reverse_bits

Write a function that takes a byte, reverses it, bit by bit (like the example) and returns the result.

Your function must be declared as follows:

```
unsigned char	reverse_bits(unsigned char octet);
```

Example:

```
 1 byte
_____________
```

```
0100  0001
  ||
  \/
1000  0010
```

24. ft_strdup

Reproduce the behavior of the function strdup (man strdup).

Your function must be declared as follows:

```
char    *ft_strdup(char *src);
```

25. max

Write the following function:

```
int                 max(int* tab, unsigned int len);
```

The first parameter is an array of int, the second is the number of elements in the array.

The function returns the largest number found in the array.

If the array is empty, the function returns 0.

26. wdmatch

Write a program that takes two strings and checks whether it's possible to
write the first string with characters from the second string, while respecting the order in which these characters appear in the second string.

If it's possible, the program displays the string, followed by a \n, otherwise it simply displays a \n.

If the number of arguments is not 2, the program displays a \n.

Examples:

```
$>./wdmatch "faya" "fgvvfdxcacpolhyghbreda" | cat -e
faya$
$>./wdmatch "faya" "fgvvfdxcacpolhyghbred" | cat -e
$
$>./wdmatch "quarante deux" "qfqfsudf arzgsayns tsregfdgs sjytdekuoixq " | cat -e
```

```
quarante deux$
$>./wdmatch "error" rrerrrfiiljdfxjyuifrrvcoojh | cat -e
$
$>./wdmatch | cat -e
$
```

27. last_word

Write a program that takes a string and displays its last word followed by a \n.

A word is a section of string delimited by spaces/tabs or by the start/end of the string.

If the number of parameters is not 1, or there are no words, display a newline.

Example:

```
$> ./last_word "FOR PONY" | cat -e
PONY$
$> ./last_word "this        ...       is sparta, then again, maybe    not" | cat -e
not$
$> ./last_word "    " | cat -e
$
$> ./last_word "a" "b" | cat -e
$
$> ./last_word "  lorem,ipsum  " | cat -e
lorem,ipsum$
$>
```

28. inter

Write a program that takes two strings and displays, without doubles, the characters that appear in both strings, in the order they appear in the first one.

The display will be followed by a \n.

If the number of arguments is not 2, the program displays \n.

Examples:

```
$>./inter "padinton" "paqefwtdjetyiytjneytjoeyjnejeyj" | cat -e
padinto$
```

$>./inter ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
df6ewg4$
$>./inter "rien" "cette phrase ne cache rien" | cat -e
rien$
$>./inter | cat -e
$

29. alpha_mirror

Write a program called alpha_mirror that takes a string and displays this string after replacing each alphabetical character by the opposite alphabetical character, followed by a newline.

'a' becomes 'z', 'Z' becomes 'A'
'd' becomes 'w', 'M' becomes 'N'

and so on.

Case is not changed.

If the number of arguments is not 1, display only a newline.

Examples:

$>./alpha_mirror "abc"
Zyx
$>./alpha_mirror "My horse is Amazing." | cat -e
Nb slihv rh Znzarmt.$
$>./alpha_mirror | cat -e
$
$>

30. do_op

Write a program that takes three strings:
The first and the third one are representations of base-10 signed integers that fit in an int.
The second one is an arithmetic operator chosen from: + - * / %

The program must display the result of the requested arithmetic operation, followed by a newline. If the number of parameters is not 3, the program just displays a newline.

You can assume the string have no mistakes or extraneous characters. Negative numbers, in input or output, will have

one and only one leading '-'. The result of the operation fits in an int.

Examples:

```
$> ./do_op "123" "*" 456 | cat -e
56088$
$> ./do_op "9828" "/" 234 | cat -e
42$
$> ./do_op "1" "+" "-43" | cat -e
-42$
$> ./do_op | cat -e
$
```

Level 03

31. rstr_capitalizer

Write a program that takes one or more strings and, for each argument, puts the last character of each word (if it's a letter) in uppercase and the rest in lowercase, then displays the result followed by a \n.

A word is a section of string delimited by spaces/tabs or the start/end of the string. If a word has a single letter, it must be capitalized.

If there are no parameters, display \n.

Examples:

```
$> ./rstr_capitalizer | cat -e
$
$> ./rstr_capitalizer "Premier PETIT TesT" | cat -e
premieR petiT tesT$
$> ./rstr_capitalizer "DeuxiEmE tEST uN PEU moinS  facile" "   attention C'EST pas dur QUAND mEmE" "ALLer UN
DeRNier 0123456789pour LA rouTE    E " | cat -e
deuxiemE tesT uN peU moinS  facilE$
attentioN c'esT paS duR quanD memE$
alleR uN dernieR 0123456789pouR lA routE    E $
$>
```

17. hidenp

Write a program named hidenp that takes two strings and displays 1 followed by a newline if the first string is hidden in the second one, otherwise displays 0 followed by a newline.

Let s1 and s2 be strings. We say that s1 is hidden in s2 if it's possible to find each character from s1 in s2, in the same order as they appear in s1.
Also, the empty string is hidden in any string.

If the number of parameters is not 2, the program displays a newline.

Examples :

```
$>./hidenp "fgex.;" "tyf34gdgf;'ektufjhgdgex.;.;rtjynur6" | cat -e
1$
$>./hidenp "abc" "2altrb53c.sse" | cat -e
1$
$>./hidenp "abc" "btarc" | cat -e
0$
$>./hidenp | cat -e
$
$>
```

18. ft_list_size

Write a function that returns the number of elements in the linked list that's passed to it.

It must be declared as follows:

```
int         ft_list_size(t_list *begin_list);
```

You must use the following structure, and turn it in as a file called
ft_list.h:

```
typedef struct    s_list
{
struct s_list *next;
 void         *data;
}              t_list;
```

19. expand_str

Write a program that takes a string and displays it with exactly three spaces between each word, with no spaces or tabs either at the beginning or the end, followed by a newline.

A word is a section of string delimited either by spaces/tabs, or by the start/end of the string.
If the number of parameters is not 1, or if there are no words, simply display a newline.

Examples:

```
$> ./expand_str "vous  voyez  c'est  facile  d'afficher  la  meme  chose" | cat -e
vous  voyez  c'est  facile  d'afficher  la  meme  chose$
$> ./expand_str " seulement        la c'est     plus dur " | cat -e
seulement  la  c'est  plus  dur$
$> ./expand_str "comme c'est cocasse" "vous avez entendu, Mathilde ?" | cat -e
$
$> ./expand_str "" | cat -e
$
$>
```

20. ft_range

Write the following function:

```
int     *ft_range(int start, int end);
```

It must allocate (with malloc()) an array of integers, fill it with consecutive values that begin at start and end at end (Including start and end !), then return a pointer to the first value of the array.

Examples:

- With (1, 3) you will return an array containing 1, 2 and 3.
- With (-1, 2) you will return an array containing -1, 0, 1 and 2.
- With (0, 0) you will return an array containing 0.
- With (0, -3) you will return an array containing 0, -1, -2 and -3.

21. add_prime_sum [No Solution yet]

Write a program that takes a positive integer as argument and displays the sum of all prime numbers inferior or equal to it followed by a newline.

If the number of arguments is not 1, or the argument is not a positive number, just display 0 followed by a newline.

Yes, the examples are right.

Examples:

```
$>./add_prime_sum 5
10
$>./add_prime_sum 7 | cat -e
17$
$>./add_prime_sum | cat -e
0$
$>
```

22. print_hex

Write a program that takes a positive (or zero) number expressed in base 10, and displays it in base 16 (lowercase letters) followed by a newline.

If the number of parameters is not 1, the program displays a newline.

Examples:

```
$> ./print_hex "10" | cat -e
a$
$> ./print_hex "255" | cat -e
ff$
$> ./print_hex "5156454" | cat -e
4eae66$
$> ./print_hex | cat -e
$
```

23. ft_rrange [No solution]

Write the following function:

```
int     *ft_rrange(int start, int end);
```

It must allocate (with malloc()) an array of integers, fill it with consecutive values that begin at end and end at start (Including start and end !), then return a pointer to the first value of the array.

Examples:

- With (1, 3) you will return an array containing 3, 2 and 1
- With (-1, 2) you will return an array containing 2, 1, 0 and -1.
- With (0, 0) you will return an array containing 0.
- With (0, -3) you will return an array containing -3, -2, -1 and 0.

24. str_capitalizer [No Solution but easy to implement, same as rstr_capitalizer but in reverse]

Write a program that takes one or several strings and, for each argument, capitalizes the first character of each word (If it's a letter, obviously),
puts the rest in lowercase, and displays the result on the standard output, followed by a \n.

A "word" is defined as a part of a string delimited either by spaces/tabs, or by the start/end of the string. If a word only has one letter, it must be capitalized.

If there are no arguments, the progam must display \n.

Example:

```
$> ./str_capitalizer | cat -e
$
$> ./str_capitalizer "Premier PETIT TesT" | cat -e
Premier Petit Test$
$> ./str_capitalizer "DeuxiEmE tEST uN PEU moinS  facile" "   attention C'EST pas dur QUAND mEmE" "ALLer UN
DeRNier 0123456789pour LA rouTE    E " | cat -e
Deuxieme Test Un Peu Moins  Facile$
 Attention C'est Pas Dur Quand Meme$
Aller Un Dernier 0123456789pour La Route    E $
$>
```

25. paramsum

Write a program that displays the number of arguments passed to it, followed by a newline.

If there are no arguments, just display a 0 followed by a newline.

Example:

```
$>./paramsum 1 2 3 5 7 24
6
$>./paramsum 6 12 24 | cat -e
3$
```

```
$>./paramsum | cat -e
0$
$>
```

26. pgcd

Write a program that takes two strings representing two strictly positive integers that fit in an int.

Display their highest common denominator followed by a newline (It's always a strictly positive integer).

If the number of parameters is not 2, display a newline.

Examples:

```
$> ./pgcd 42 10 | cat -e
2$
$> ./pgcd 42 12 | cat -e
6$
$> ./pgcd 14 77 | cat -e
7$
$> ./pgcd 17 3 | cat -e
1$
$> ./pgcd | cat -e
$
```

Level 04

27. fprime

Write a program that takes a positive int and displays its prime factors on the standard output, followed by a newline.

Factors must be displayed in ascending order and separated by '*', so that the expression in the output gives the right result.

If the number of parameters is not 1, simply display a newline.

The input, when there's one, will be valid.

Examples:

```
$> ./fprime 225225 | cat -e
3*3*5*5*7*11*13$
$> ./fprime 8333325 | cat -e
3*3*5*5*7*11*13*37$
$> ./fprime 9539 | cat -e
9539$
$> ./fprime 804577 | cat -e
804577$
$> ./fprime 42 | cat -e
2*3*7$
$> ./fprime 1 | cat -e
1$
$> ./fprime | cat -e
$
$> ./fprime 42 21 | cat -e
$
```

28. ft_list_remove_if

Write a function called ft_list_remove_if that removes from the passed list any element the data of which is "equal" to the reference data.

It will be declared as follows :

```
void ft_list_remove_if(t_list **begin_list, void *data_ref, int (*cmp)());
```

cmp takes two void* and returns 0 when both parameters are equal.

You have to use the ft_list.h file, which will contain:

```
$>cat ft_list.h
typedef struct      s_list
{
 struct s_list   *next;
void            *data;
}                t_list;
$>
```

29. sort_int_tab

Write the following function:

void sort_int_tab(int *tab, unsigned int size);

It must sort (in-place) the 'tab' int array, that contains exactly 'size' members, in ascending order.

Doubles must be preserved.

Input is always coherent.

30. rev_wstr

Write a program that takes a string as a parameter, and prints its words in reverse order.

A "word" is a part of the string bounded by spaces and/or tabs, or the
begin/end of the string.

If the number of parameters is different from 1, the program will display '\n'.

In the parameters that are going to be tested, there won't be any "additional" spaces (meaning that there won't be additionnal spaces at the beginning or at the end of the string, and words will always be separated by exactly one space).

Examples:

```
$> ./rev_wstr "le temps du mepris precede celui de l'indifference" | cat -e
l'indifference de celui precede mepris du temps le$
$> ./rev_wstr "abcdefghijklm"
Abcdefghijklm
$> ./rev_wstr "il contempla le mont" | cat -e
mont le contempla il$
$> ./rev_wstr | cat -e
$
$>
```

31. tab_mult

Write a program that displays a number's multiplication table.

The parameter will always be a strictly positive number that fits in an int, and said number times 9 will also fit in an int.

If there are no parameters, the program displays \n.

Examples:

```
$>./tab_mult 9
1 x 9 = 9
2 x 9 = 18
3 x 9 = 27
4 x 9 = 36
5 x 9 = 45
6 x 9 = 54
7 x 9 = 63
8 x 9 = 72
9 x 9 = 81
$>./tab_mult 19
1 x 19 = 19
2 x 19 = 38
3 x 19 = 57
4 x 19 = 76
5 x 19 = 95
6 x 19 = 114
7 x 19 = 133
8 x 19 = 152
9 x 19 = 171
$>
$>./tab_mult | cat -e
$
$>
```

32. sort_list [No solution yet]

Write the following functions:

```
t_list      *sort_list(t_list* lst, int (*cmp)(int, int));
```

This function must sort the list given as a parameter, using the function pointer cmp to select the order to apply, and returns a pointer to the first element of the sorted list.

Duplications must remain.

Inputs will always be consistent.

You must use the type t_list described in the file list.h that is provided to you. You must include that file (#include "list.h"), but you must not turn it in. We will use our own to compile your assignment.

Functions passed as cmp will always return a value different from 0 if a and b are in the right order, 0 otherwise.

For example, the following function used as cmp will sort the list in ascending order:

```
int ascending(int a, int b)
{
        return (a <= b);
}
```

33. ft_list_foreach

Write a function that takes a list and a function pointer, and applies this function to each element of the list.

It must be declared as follows:

```
void    ft_list_foreach(t_list *begin_list, void (*f)(void *));
```

The function pointed to by f will be used as follows:

```
(*f)(list_ptr->data);
```

You must use the following structure, and turn it in as a file called ft_list.h:

```
typedef struct    s_list
{
 struct s_list *next;
void          *data;
}
```

Level 05

34. ft_atoi_base

Write a function that converts an integer value to a null-terminated string using the specified base and stores the result in a char array that you must allocate.

The base is expressed as an integer, from 2 to 16. The characters comprising the base are the digits from 0 to 9, followed by uppercase letter from A to F.

For example, base 4 would be "0123" and base 16 "0123456789ABCDEF".

If base is 10 and value is negative, the resulting string is preceded with a minus sign (-). With any other base, value is always considered unsigned.

Your function must be declared as follows:

```
char	*ft_itoa_base(int value, int base);
```

35. brainfuck

Write a Brainfuck interpreter program.
The source code will be given as first parameter.
The code will always be valid, with no more than 4096 operations.
Brainfuck is a minimalist language. It consists of an array of bytes (in our case, let's say 2048 bytes) initialized to zero, and a pointer to its first byte.

Every operator consists of a single character :
- '>' increment the pointer ;
- '<' decrement the pointer ;
- '+' increment the pointed byte ;
- '-' decrement the pointed byte ;
- '.' print the pointed byte on standard output ;
- '[' go to the matching ']' if the pointed byte is 0 (while start) ;
- ']' go to the matching '[' if the pointed byte is not 0 (while end).

Any other character is a comment.

Examples:

```
$>./brainfuck "++++++++++[>+++++++>++++++++++>+++>+<<<<-]
```

```
>++.>+.+++++++..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.>." | cat -e
Hello World!$
$>./brainfuck "+++++[>++++[>++++H>+++++i<<-]>>>++\n<<<<-]>>--------.>+++++.>." | cat -e
Hi$
$>./brainfuck | cat -e
$
```

36. print_memory

Write a function that takes (const void *addr, size_t size), and displays the memory as in the example.

Your function must be declared as follows:

```
void	print_memory(const void *addr, size_t size);
```

```
$> cat main.c
void	print_memory(const void *addr, size_t size);

int	main(void)
{
	int	tab[10] = {0, 23, 150, 255, 12, 16,  21, 42};

	print_memory(tab, sizeof(tab));
	return (0);
}
$> gcc -Wall -Wall -Werror main.c print_memory.c && ./a.out | cat -e
0000 0000 1700 0000 9600 0000 ff00 0000 ................$
0c00 0000 1000 0000 1500 0000 2a00 0000 ............*...$
0000 0000 0000 0000                     ........$
```

37. str_maxlenoc [No solution]

Write a program that takes one or more strings and displays, followed by a newline, the longest string that appears in every parameter. If more that one string qualifies, it will display the one that appears first in the first parameter. Note that the empty string technically appears in any string.

If there are no parameters, the program displays \n.

Examples:

```
$>./str_maxlenoc ab bac abacabccabcb
A
$>./str_maxlenoc bonjour salut bonjour bonjour
U
$>./str_maxlenoc xoxAoxo xoxAox  oxAox oxo  A ooxAoxx oxooxo Axo | cat -e
$
$>./str_maxlenoc bosdsdfnjodur atehhellosd afkuonjosurafg headfgllosf fghellosag afdfbosnjourafg
Os
$>./str_maxlenoc | cat -e
```

38. ord_alphlong [No solution]

Write a program that takes a string as a parameter and prints its words sorted in length order then in alphabetical order: when words are alphabetically equals (for example aA and Aa), original order must remain (lower and upper case are the same in alphabetical order). If there are duplicates, they must remain.

If number of parameters is different from 1, the program prints \n.

There will be only spaces, tabs and alphanumeric caracters in strings.

You'll print only one space between each word. Nothing before the first and after the last word of each line.

Examples:

```
$>./ord_alphlong
$
$>./ord_alphlong "De son baton il frappa la pierre et l eau jaillit" | cat -e
l$
De et il la$
eau son$
baton$
frappa pierre$
jaillit$
$>./ord_alphlong "A a b B cc ca cd" | cat -e
A a b B$
ca cc cd$
$>./ord_alphlong "Pour l Imperium de l humanite" | cat -e
l l$
de$
```

Pour$
humanite Imperium$
$>

Unknown Level

39. union [no solution]

Write a program that takes two strings and displays, without doubles, the characters that appear in either one of the strings.

The display will be in the order characters appear in the command line, and will be followed by a \n.

If the number of arguments is not 2, the program displays \n.

Example:

$>./union zpadinton "paqefwtdjetyiytjneytjoeyjnejeyj" | cat -e
zpadintoqefwjy$
$>./union ddf6vewg64f gtwthgdwthdwfteewhrtag6h4ffdhsd | cat -e
df6vewg4thras$
$>./union "rien" "cette phrase ne cache rien" | cat -e
rienct phas$
$>./union | cat -e
$
$>
$>./union "rien" | cat -e
$
$>

40. count_alpha [no solution]

Write a program called count_alpha that takes a string and displays the number of occurences of its alphabetical characters. Other characters are not counted.
The order is the order of occurrence in the string. The display must be ended by a newline.

The format is in the examples.

If the number of arguments is not 1, display only a newline.

Examples :

```
$> ./count_alpha abbcc
1a, 2b, 2c
$> ./count_alpha "abbcc"
1a, 2b, 2c
$> ./count_alpha "abbcc" "dddeef" | cat -e
$
$> ./count_alpha "My Hyze 47y 7." | cat -e
1m, 3y, 1h, 1z, 1e$
$> ./count_alpha "" | cat -e
$
```