**EXAM02: 7 exercices**

**0-aff\_a or 0-aff\_z …………………………………………………………………………………………………………………………14points**

**1-rev\_print or 1-ft\_swap or 1-ft-strlen ………………………………………………………………… 28/100**

**2-first\_word or 2-ft\_strrev or 2-rot\_13 or 2-rotone…………………………………………….42/100**

**3-inter or 3-last\_word or 3-union or 3-wdmatch ……………………………………………………. 56/100**

**4-ft\_atoi ………………………………………………………………………………………………………………………………………………….. 70/100**

**5-ft\_range …………………………………………………………………………………………………………………………………………………… 84/100**

**6-ft\_itoa or 6-ft\_split ……………………………………………………………………………………………………………. 100/100**

**Answer: 00-A**

**#include <unistd.h>**

**int     main(int ac, char \*\*av)**

**{**

**int i;**

**i = 0;**

**if (ac == 2)**

**{**

**while (av[1][i] != '\0')**

**{**

**if (av[1][i] == 'a')**

**{**

**write(1, "a", 1);**

**break ;**

**}**

**i++;**

**}**

**write(1, "\n", 1);**

**}**

**else**

**write(1, "a\n", 2);**

**return (0);**

**}**

**Answer: 00-B**

**#include <unistd.h>**

**int     main(int ac, char \*\*av)**

**{**

**(void)ac;**

**(void)av;**

**write(1, "z\n", 2);**

**return (0);**

**}**

**Answer: 01-A**

**Assignment name : ft\_strlen**

**Expected files : ft\_strlen.c**

**Allowed functions:**

**--------------------------------------------------------------------------------**

**Write a function that returns the length of a string.**

**Your function must be declared as follows:**

**int ft\_strlen(char \*str);**

**SOLUTION:**

**int     ft\_strlen(char \*str)**

**{**

**int i;**

**i = 0;**

**while (str[i])**

**i++;**

**return (i);**

**}**

**Answer: 01-B**

**Assignment name : ft\_swap**

**Expected files : ft\_swap.c**

**Allowed functions:**

**--------------------------------------------------------------------------------**

**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);**

**SOLUTION:**

**void    ft\_swap(int \*a, int \*b)**

**{**

**int tmp;**

**tmp = \*a;**

**\*a = \*b;**

**\*b = tmp;**

**}**

**Answer: 01-C**

**Assignment name : rev\_print**

**Expected files : rev\_print.c**

**Allowed functions: write**

**--------------------------------------------------------------------------------**

**Write a function that takes a string and displays the string in reverse order followed by the newline. Its prototype is constructed like this :**

**char \*ft\_rev\_print (char \*str)**

**It must return its argument**

**Examples:**

**$> ./rev\_print "zaz" | cat -e**

**zaz$**

**$> ./rev\_print "dub0 a POIL" | cat -e**

**LIOP a 0bud$**

**$> ./rev\_print | cat -e**

**$**

**SOLUTION:**

**#include <unistd.h>**

**#include <stdio.h>**

**int ft\_strlen(char \*str)**

**{**

**int i;**

**i = 0;**

**while(str[i] != '\0')**

**i++;**

**return (i);**

**}**

**char \*ft\_rev\_print(char \*str)**

**{**

**int i;**

**i = ft\_strlen(str);**

**i--;**

**while (i >= 0)**

**{**

**write(1, &str[i], 1);**

**i--;**

**}**

**return(str);**

**}**

**int main()**

**{**

**char str[] = "gbrireugneb";**

**ft\_rev\_print(str);**

**}**

**Answer: 02-A**

**Assignment name : first\_word**

**Expected files : first\_word.c**

**Allowed functions: write**

**--------------------------------------------------------------------------------**

* **Write a program that takes a string and displays its first word, followed by a newline.**
* **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 if there are no words, simply display a newline.**

**Examples:**

**$> ./first\_word "FOR PONY" | cat -e**

**FOR$**

**$> ./first\_word "this ... is sparta, then again, maybe not" | cat -e**

**this$**

**$> ./first\_word " " | cat -e**

**$**

**$> ./first\_word "a" "b" | cat -e**

**$**

**$> ./first\_word " lorem,ipsum " | cat -e**

**lorem,ipsum$**

**$>**

**SOLUTION:**

**#include <unistd.h>**

**int     main(int ac, char \*\*av)**

**{**

**int i;**

**i = 0;**

**if (ac == 2)**

**{**

**while (av[1][i] == ' ' || av[1][i] == '\t')**

**i++;**

**while (av[1][i] != '\0' && av[1][i] != ' ' && av[1][i] != '\t')**

**{**

**write(1, &av[1][i], 1);**

**i++;**

**}**

**}**

**write(1, "\n", 1);**

**return (0);**

**}**

**Answer: 02-B**

**Assignment name : ft\_strrev**

**Expected files : ft\_strrev.c**

**Allowed functions:**

**--------------------------------------------------------------------------------**

**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);**

**SOLUTION:**

**int     ft\_strlen(char \*str)**

**{**

**int i;**

**i = 0;**

**while (str[i] != '\0')**

**i++;**

**return (i);**

**}**

**char    \*ft\_strrev(char \*str)**

**{**

**int i;**

**int len;**

**char tmp;**

**i = 0;**

**len = ft\_strlen(str) - 1;**

**while (len > i)**

**{**

**tmp = str[i];**

**str[i] = str[len];**

**str[len] = tmp;**

**i++;**

**len--;**

**}**

**return (str);**

**}**

**Answer: 02-C**

**Assignment name : rot\_13**

**Expected files : rot\_13.c**

**Allowed functions: write**

**--------------------------------------------------------------------------------**

**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**

**$**

**$>**

**SOLUTION:**

**#include <unistd.h>**

**void    rot13(char \*str)**

**{**

**int i;**

**i = 0;**

**while( str[i] != '\0')**

**{**

**if( (str[i] >= 'A' && str[i] <= 'M') || (str[i] >= 'a' && str[i] <= 'm') )**

**str[i] += 13;**

**else if( (str[i] >= 'N' && str[i] <= 'Z') || (str[i] >= 'n' && str[i] <= 'z') )**

**str[i] -= 13;**

**write(1, &str[i], 1);**

**i++;**

**}**

**}**

**int main(int ac, char \*\*av)**

**{**

**if (ac == 2)**

**rot13(av[1]);**

**write(1, "\n", 1);**

**}**

**Answer: 02-D**

**Assignment name : rotone**

**Expected files : rotone.c**

**Allowed functions: write**

**--------------------------------------------------------------------------------**

**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**

**$**

**$>**

**SOLUTION:**

**#include <unistd.h>**

**void    rotone(char \* str)**

**{**

**int i;**

**i = 0;**

**while(str[i] != '\0')**

**{**

**if ( (str[i] >= 'A' && str[i] <= 'Y') || (str[i] >= 'a' && str[i] <= 'y'))**

**str[i] += 1;**

**else if (str[i] == 'Z' || str[i] == 'z')**

**str[i] -= 25;**

**write (1, &str[i], 1);**

**i++;**

**}**

**}**

**int main(int ac, char \*\*av)**

**{**

**if (ac == 2)**

**rotone(av[1]);**

**write(1, "\n", 1);**

**}**

**Answer: 03-A**

**Assignment name : inter**

**Expected files : inter.c**

**Allowed functions: write**

**--------------------------------------------------------------------------------**

**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**

**$**

**SOLUTION:**

**#include <unistd.h>**

**int     check\_doubles(char \*str, char c, int pos)**

**{**

**int i;**

**i = 0;**

**while (i < pos)**

**{**

**if (str[i] == c)**

**return (0);**

**i++;**

**}**

**return (1);**

**}**

**void    inter(char \*str, char \*str1)**

**{**

**int i;**

**int j;**

**i = 0;**

**while (str[i] != '\0')**

**{**

**j = 0;**

**while (str1[j] != '\0')**

**{**

**if (str[i] == str1[j])**

**{**

**if (check\_doubles(str, str[i], i) == 1)**

**{**

**write(1, &str[i], 1);**

**break;**

**}**

**}**

**j++;**

**}**

**i++;**

**}**

**}**

**int     main(int ac, char \*\*av)**

**{**

**if (ac == 3)**

**inter(av[1], av[2]);**

**write(1, "\n", 1);**

**return (0);**

**}**

**Answer: 03-B**

**Assignment name : last\_word**

**Expected files : last\_word.c**

**Allowed functions: write**

**--------------------------------------------------------------------------------**

**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$**

**$>**

**SOLUTION:**

**#include <unistd.h>**

**void    last\_word(char \*str)**

**{**

**int i = 0;**

**while (str[i] != '\0')**

**i++;**

**i -= 1;**

**while(str[i] == '\t' || str[i] == 32)**

**i--;**

**while (i > 0)**

**{   if(str[i] == 32 && str[i] == '\t')**

**break;**

**i--;**

**}**

**i++;**

**while (str[i] != '\0' && str[i] != 32 && str[i] != '\t')**

**{**

**write(1, &str[i], 1);**

**i++;**

**}**

**}**

**int main(int ac, char \*\*av)**

**{**

**if (ac == 2)**

**last\_word(av[1]);**

**write(1, "\n", 1);**

**}**

**Answer: 03-C**

**Assignment name : union**

**Expected files : union.c**

**Allowed functions: write**

**--------------------------------------------------------------------------------**

**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**

**$**

**$>**

**SOLUTION:**

**#include <unistd.h>**

**int     check\_doubles2(char \*str, char c)**

**{**

**int i;**

**i = 0;**

**while (str[i] != '\0')**

**{**

**if (str[i] == c)**

**return (0);**

**i++;**

**}**

**return (1);**

**}**

**int     check\_doubles1(char \*str, char c, int pos)**

**{**

**int i;**

**i = 0;**

**while (i < pos)**

**{**

**if (str[i] == c)**

**return (0);**

**i++;**

**}**

**return (1);**

**}**

**void    ft\_union(char \*str, char \*str1)**

**{**

**int i;**

**i = 0;**

**while (str[i] != '\0')**

**{**

**if (check\_doubles1(str, str[i], i) == 1)**

**write(1, &str[i], 1);**

**i++;**

**}**

**i = 0;**

**while (str1[i] != '\0')**

**{**

**if (check\_doubles2(str, str1[i]) == 1)**

**{**

**if (check\_doubles1(str1, str1[i], i) == 1)**

**write(1, &str1[i], 1);**

**}**

**i++;**

**}**

**}**

**int     main(int ac, char \*\*av)**

**{**

**if (ac == 3)**

**ft\_union(av[1], av[2]);**

**write(1, "\n", 1);**

**return (0);**

**}**

**Answer: 03-D**

**Assignment name : wdmatch**

**Expected files : wdmatch.c**

**Allowed functions: write**

**----------------------------------------------------**

**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**

**$**

**SOLUTION:**

**#include <unistd.h>**

**void    ft\_putstr(char \*str)**

**{**

**int i;**

**i = 0;**

**while (str[i] != '\0')**

**{**

**write(1, &str[i], 1);**

**i++;**

**}**

**}**

**int     ft\_strlen(char \*str)**

**{**

**int i;**

**i = 0;**

**while (str[i] != '\0')**

**i++;**

**return (i);**

**}**

**int     main(int ac, char \*\*av)**

**{**

**int i;**

**int j;**

**int wdlen;**

**i = 0;**

**j = 0;**

**wdlen = 0;**

**if (ac == 3)**

**{**

**while (av[1][i] != '\0')**

**{**

**while (av[2][j] != '\0')**

**{**

**if (av[1][i] == av[2][j])**

**{**

**wdlen++;**

**break ;**

**}**

**j++;**

**}**

**i++;**

**}**

**if (wdlen == ft\_strlen(av[1]))**

**ft\_putstr(av[1]);**

**}**

**write(1, "\n", 1);**

**return (0);**

**}**

**Answer: 04**

**Assignment name : ft\_atoi**

**Expected files : ft\_atoi.c**

**Allowed functions: None**

**----------------------------------------------------------------------**

**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);**

**int    ft\_atoi(const char \*str)**

**{**

**int result;**

**int sign;**

**result = 0;**

**sign = 1;**

**while (\*str == 32 || (\*str >= 9 && \*str <= 13))**

**str++;**

**while (\*str == '-' || \*str == '+')**

**{**

**if (\*str == '-')**

**{**

**sign \*= -1;**

**}**

**str++;**

**}**

**while (\*str >= '0' && \*str <= '9')**

**{**

**result = result \* 10 + (\*str - '0');**

**str++;**

**}**

**return (sign \* result);**

**}**

**#include <stdio.h>**

**int main (void)**

**{**

**char \*str;**

**str = "   ---+--+12345ab67";**

**printf ("%d", ft\_atoi (str));**

**}**

**Answer: 05**

**Assignment name : ft\_range**

**Expected files : ft\_range.c**

**Allowed functions: malloc**

**--------------------------------------------------------------------------------**

**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.**

**SOLUTION:**

**#include <stdlib.h>**

**#include <stdio.h>**

**int        ft\_abs(int x)**

**{**

**if (x < 0)**

**return (-x);**

**return (x);**

**}**

**int    \*ft\_range(int start, int end)**

**{**

**int        size;**

**int        i;**

**int        \*tab;**

**int        \*d;**

**size = ft\_abs(end - start) + 1;**

**d = (tab = malloc(size \* sizeof(int)));**

**if (!d)**

**return (0);**

**i = 0;**

**if ( size == 1)**

**tab[0] = start;**

**if (start < end)**

**{**

**while (i < size)**

**{**

**tab[i] = start + i;**

**i++;**

**}**

**}**

**else if (start > end)**

**{**

**while (i < size)**

**{**

**tab[i] = start - i;**

**i++;**

**}**

**}**

**return (tab);**

**}**

**int        main(void)**

**{**

**int \*tab;**

**int i = 0;**

**int start = 0;**

**int end = 0 ;**

**int size = ft\_abs(end - start) + 1;**

**tab = ft\_range(start, end);**

**while(i < size)**

**{**

**printf("%i, ", tab[i]);**

**i++;**

**}**

**}**

**Answer: 06-A**

**Assignment name : ft\_itoa**

**Expected files : ft\_itoa.c**

**Allowed functions: malloc**

**--------------------------------------------------------------------------------**

**Write a function that takes an int and converts it to a null-terminated string.**

**The function returns the result in a char array that you must allocate.**

**Your function must be declared as follows:**

**char \*ft\_itoa(int nbr);**

**SOLUTION:**

**#include <stdlib.h>**

**#include <stdio.h>**

**int len(long nb)**

**{**

**int len = 0;**

**if (nb < 0)**

**{**

**nb \*= -1;**

**len++;**

**}**

**while (nb > 0)**

**{**

**nb /= 10;**

**len++;**

**}**

**return(len);**

**}**

**char    \*ft\_itoa(int nb)**

**{**

**char    \*str;**

**long    n;**

**int i;**

**n = nb;**

**i = len(n);**

**if(!(str = (char \*)malloc(i + 1)))**

**return(0);**

**str[i--] = '\0';**

**if (n == 0)**

**{**

**str[0] = 48;**

**return(str);**

**}**

**if(n < 0)**

**{**

**str[0] = '-';**

**n \*= -1;**

**}**

**while (n > 0)**

**{**

**str[i] = 48 + (n % 10);**

**n /= 10;**

**i--;**

**}**

**return (str);**

**}**

**int main(void)**

**{**

**printf("%s\n", ft\_itoa(1342345));**

**}**

**int main(void)**

**A computer screen shot of a computer screen

Description automatically generated**

**char    \*ft\_itoa(int nb)**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**int len(long nb)**

**A screenshot of a computer

Description automatically generatedA close-up of a number

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA close-up of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA close-up of a computer screen

Description automatically generated**

**\*ft\_itoa(int nb) ….. i = len(n);**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated…….A screenshot of a computer

Description automatically generated**

**int main(void) >>>>>>>>     printf("%s\n", ft\_itoa(1342345));**

**A screenshot of a computer

Description automatically generated**

**Answer: 06-B**

**Assignment name : ft\_split**

**Expected files : ft\_split.c**

**Allowed functions: malloc**

**--------------------------------------------------------------------------------**

**Write a function that takes a string, splits it into words, and returns them as a NULL-terminated array of strings.**

**A "word" is defined as a part of a string delimited either by spaces/tabs/new lines, or by the start/end of the string.**

**Your function must be declared as follows:**

**char \*\*ft\_split(char \*str);**

**SOLUTION:**

**#include <stdlib.h>**

**int check\_separator(char c)**

**{**

**if ( c == 10 || c == 9 || c == 32)**

**return (1);**

**if (c == 0)**

**return (1);**

**return (0);**

**}**

**int count\_strings(char \*str)**

**{**

**int i;**

**int count;**

**count = 0;**

**i = 0;**

**while (str[i] != '\0')**

**{**

**while (str[i] != '\0' && check\_separator(str[i]))**

**i++;**

**if (str[i] != '\0')**

**count++;**

**while (str[i] != '\0' && !check\_separator(str[i]))**

**i++;**

**}**

**return (count);**

**}**

**int ft\_strlen\_sep(char \*str)**

**{**

**int i;**

**i = 0;**

**while (str[i] && !check\_separator(str[i]))**

**i++;**

**return (i);**

**}**

**char    \*ft\_word(char \*str)**

**{**

**int     len\_word;**

**int     i;**

**char    \*word;**

**i = 0;**

**len\_word = ft\_strlen\_sep(str);**

**word = (char \*)malloc(sizeof(char) \* (len\_word + 1));**

**while (i < len\_word)**

**{**

**word[i] = str[i];**

**i++;**

**}**

**word[i] = '\0';**

**return (word);**

**}**

**char    \*\*ft\_split(char \*str)**

**{**

**char    \*\*strings;**

**int     i;**

**i = 0;**

**strings = (char \*\*)malloc(sizeof(char \*)**

**\* (count\_strings(str) + 1));**

**while (\*str != '\0')**

**{**

**while (\*str != '\0' && check\_separator(\*str))**

**str++;**

**if (\*str != '\0')**

**{**

**strings[i] = ft\_word(str);**

**i++;**

**}**

**while (\*str && !check\_separator(\*str))**

**str++;**

**}**

**strings[i] = 0;**

**return (strings);**

**}**

**#include <stdio.h>**

**int main(int argc, char \*\*argv)**

**{**

**int     index;**

**char    \*\*split;**

**(void)  argc;**

**split = ft\_split(argv[1]);**

**index = 0;**

**while (split[index])**

**{**

**printf("%s\n", split[index]);**

**index++;**

**}**

**}**