C1- Unix & C Lab Seminar

pool_c_d12

Day 12

For Break dance, Continue to Switch

CODING ACADEMY
BY EPITECH

Instructions

Before You Go Further

- Turn in directory **pool_c_d12**.
- Respecting the norm takes time, but it is good for you. This way your code will respect the norm from the first written line to the last.
  Read carefully the norm documentation. You should type "alt+i" instead of "tab"
- You shall leave in your directory no other files than those explicitly specified by the exercices. If one of your files prevents the compilation with *.c, the robot will not be able to do the correction and you will have a 0.
- **Do not turn-in a main() function unless it has been explicitly asked.**

Remember it is always better to create your repository at the beginning of the day and to turn-in your work on a regular basis. Do not forget the permissions rights!

Introduction

Starting from today, you do not need to respect the norm anymore. This doesn't mean that you can freely make the ugliest code because we will keep the right to refuse to answer a question if your code is too ugly. Moreover during a "Soutenance" you could lose points if we think that the code is too "dirty".

However, this means that you do not have any forbidden keyword or functions from now on. You should take a look at the keyword "for", "break", "switch" and "continue", those one are the most common ones.

Exercise 1

Print Unique (2pts)

Turn in: pool_c_d12/ex_01/print_unique.c
Prototype: void print_unique(int *array, int size);

Write a function 'print_unique' taking an array of int as it first parameter and the size of this array as it second parameter, this function will print every number that are unique in the array followed by a comma, except for the last which will be followed by a new line.

Example :
main.c :

```
void    print_unique(int *, int);
int     main()
{
  int   tab[] = {0, 0, 2, 3, 4, 3, 6, 1};

  print_unique(tab, 8);
  return (0);
}
```

Terminal

```
~/pool_c_d12> cc *.c -o ex01
~/pool_c_d12> ./ex01
2,4,6,1
~/pool_c_d12>
```

The use of "break" or "continue" may be useful here but is not a necessity

Exercise 2

Merge Array (3pts)

Turn in: pool_c_d12/ex_02/merge_array.c
Prototype: int *merge_array(int *arr1, int *arr2, int size1, int size2);

Write a function 'merge_array' taking four parameters, the first two being two array of int and the second two being their respective size.
The function will return a newly allocated array containing a merge of the content of the two arrays passed as parameters.

Exercise 3

Magic Square (3pts)

Turn in: pool_c_d12/ex_03/magic_square.c
Prototype: int magic_square(int *sqr);

Write a function 'magic_square' taking an array of int as it first parameter. This array will always have a size of nine. This function will return 0 if this array is a valid magic square and 1 otherwise.
A magic square is a square in which the sum of the numbers in each line is equal to the sum of numbers in each column and each digonals.

Example :
8 1 6
3 5 7
4 9 2
Is a magic square where the sums are always equal to 15.

Your function will receive these numbers that will already be in an array like this :
N0 N1 N2 N3 N4 N5 N6 N7 N8
The magic square that you need to test is then :
N0 N1 N2
N3 N4 N5
N6 N7 N8

Exercise 4

Split Array (3pts)

Turn in: pool_c_d12/ex_04/split_array.c
Prototype: int **split_array(int *arr, int size, int *new_size1, int *new_size2);

Write a function 'split_array'. It will take an array as it first parameter and a size as it second parameter. It will also takes two pointers to int. This function will return a pointer to two arrays of int. This first of this two array will containt all the odd numbers contained in the array passed as parameter, the second one will contain all the even numbers contained in the array passed as parameter.

Your function must store the size of the new array in the pointers to int given as parameters. The size of the odds array will be stored in "new_size1" and the size of the evens one will be stored in "new_size2".

Exercise 5

Pascal Triangle (4pts)

Turn in: pool_c_d12/ex_05/pascalTr.c
Prototype: int **pascalTr(int);

Write a function 'pascalTr' that will take a size as it first parameter and return a two-dimensional array dynamically allocated representing a Pascal triangle.
A pascal triangle is a triangle where the first and second rows are set to 1, each element starting from the third row is the sum of the element directly above it and the element to the left of the element directly above it.

Example with a Pascal Triangle of size 6:

```
1
1       1
1       2       1
1       3       3       1
1       4       6       4       1
1       5       10      10      5       1
```

Exercise 6

Matrices Addition (5pts)

Turn in: pool_c_d12/ex_06/matrices.c
Prototype: int matrices_addition(int **, int column_offset, int row_offset, int size, int direction);

Write a function named 'matrices_addition' taking five parameters. The first one will be a double array of int, the second will be the offset you will need to use for the column, which mean the starting column for our addition, the third one will be the same but for the row. The fourth parameter will be the size of the matrices, **we will only use square matrices.** And finally, the fifth parameter will be the "direction" we should move inside, we will represent it with an int. There will be a total of 8 ways, taking values corresponding to a clock, for example if the value is 3 you will need to go to the right inside the matrices, for diagonals we should be able to use both values (see example.).

Example :

```
10/11   0       1/2
9               3
7/8     6       4/5
```

This function must return the sum of the elements chosen.
If **any** error occurs the function will simply return 0. main.c

```c
#include <stdio.h>

int     matrices_addition(int **, int column_offset, int row_offset,    \
                          int size, int direction);

int     main()
{
  int   mat[5][5] =
    {
      {0, 1, 2, 3, 4},
      {5, 6, 7, 8, 9},
      {10, 11, 12, 13, 14},
      {15, 16, 17, 18, 19},
      {20, 21, 22, 23, 24},
    };
  int   *tab[5];

  tab[0] = mat[0];
  tab[1] = mat[1];
  tab[2] = mat[2];
  tab[3] = mat[3];
  tab[4] = mat[4];
  printf("%d\n", matrices_addition(tab, 0, 0, 5, 3));
  printf("%d\n", matrices_addition(tab, 0, 0, 5, 6));
  printf("%d\n", matrices_addition(tab, 0, 0, 5, 0));
  printf("%d\n", matrices_addition(tab, 3, 3, 5, 11));
  printf("%d\n", matrices_addition(tab, 3, 3, 5, -2));
  printf("%d\n", matrices_addition(tab, 3, 22, 5, 2));
  return (0);
}
```

```
Terminal
~/pool_c_d12> cc *.c -o ex05
~/pool_c_d12> ./ex05
10
50
0
36
0
0
~/pool_c_d12>
```

Exercise Bonus

White Rabbit (2pts)

Turn in: pool_c_d12/ex_07/white_rabbit.c
Prototype: int follow_the_white_rabbit();

Sitting on top of the hill, Alice was bored to death, wondering if making a chaplet of flowers was worth getting up and actually gathering said flowers. And then, suddently, a White Rabbit with pink eyes passed by, running like a madman. This was actually not really worth a mention, and Alice wasn't much more puzzled to head the Rabbit mumbling : "Oh my god, Oh my god ! I'm going to be late !". However, from the moment the Rabbit pulled out a watch from the pocket of his vest, looked at the time, and ran even faster, Alice was on her feet in a heartbeat. She suddenly realized that she had never seen either a Rabbit with a vest pocket, nor a watch to pull out from there. Dying to know more, she ran through the fields in the rabbit's wake, and was lucky enough to be right on time to see him rushing in to a huge burrow, under the bushed. Not a moment later was she already inside, never even wondering how the hell she would get out of there.

After drifting around for a long, long time, in the maze of the burrow's walls, Alice met a Gecko whose words, more or less, were these: "Hey there ye ! Wot's you doin' here ? Lookin' fer the pink pony as well ?". Not a pink pony but a white rabbit, Alice answered. "Aaah, but I know him well, th'old rabbit friend", the Gecko retorted. "I even saw him not five minutes ago ! 'Looked like he was in a hell outta an hurry, th'old rabbit friend !". Alice asked the Gecko to show her the direction the rabbit was heading. Without an hesitation, the Gecko pointed to his left and blurted out : "Thatta way !!", before suddenly pausing and pointing to the opposite direction. "Err, nay... I think it wos rather thatta way...". After having pointed to a dozen of different directions, the Gecko finally admitted "Hmmm... Actually, I think I may well be lost.". Alice was in despair. She was list in a huge burrow, and was ofd the trail of the white rabbit. When he saw her in this sate, the Gecko took pity on her, and told her : "Dun' worry there gal, we'll find your friend th'White Rabbit. Look what I got there.". He immediatly took a 37-faced dice out of his vest pocket (yes, he also has a vest pocket) and handed it to Alice. He showed each of the faces to Alice. "This is the first face - yeh can tell cause o'the number 1 written on it. And this is face 2", and so on until reaching the 37^{th} one.

The Gecko then told Alice : "What yeh got in yer hand, it's a magic dice ! Yeh must take real care of it ! But it'll make yeh find the White Rabbit ! Now, listen well, open yer hearsm I'll explain to yeh how yeh must use it. Each time yeh don't know where t'go anymore, throw thees dice.
Th'result'll tell yeh which direction the White Rabbit took. Although, if the dice gives yeh a multiple of 11 and the weather is nice, y'should always take a nap - might as well enjoy the sun. 'Cause yeh can be sure the White Rabbit'll do the same. But if the weather is crap, better launch the dice again. Same thing whene you wake up after the nap, 'f'course. If the weather is still nice it tells ye to nap again, you woke up waaay too early ! If you get a 37, then you found the White Rabbit. Never forget that after a cup of tea, you should always go straight ahead. When you get a face that's higher than 24 and that three times this face gives you seven-times-ten-eight or 146, means the dice was wrong. y'should turn and head back. If the result is four or 5, go left, there's a caterpillar here that sucks on that pipe of hers all day long and makes circles with her smoke. Bit crazy she is, but quite funneh ! With a 15, straight ahead with you. When the dice says 12, ye're out of luck, that was for naught and yeh have to throw again. When th'result is 13, head to yer right. Also works with 34 or more. Left it is, if the dice says six, 17 or 18. A 23 means it's 10pm ! Time for a cup of tea. Find a table, and order a lemon tea. If you don't like lemon, green tea is fine enough. Oh right, never forget to count all your results ! When you add'em and yeh find 421, yeh found the White Rabbit. Say hi for me, while you're at it. Oh, I need to tell you : that count the result thing, that also works with three-times-hundred-and-ninety-seven at least. Whenever you get a result that you can divide by 8 and get a round result with nothing left, just head back where you came from. That number 8 is crappy, I don't like it. When the result is twice of three times 5, keep going ahead, yer on the good way ! The sum though is quite a bugger, if the dice tells you you found the White Rabbit, y'still need to do what the dice told tou to do ! If it tell yeh t'go left, well y'go left and th'White Rabbit'll be there. If it tells right, then you don't go left, you go right ! Well, you got it anyway. Dun worry, everything'll be fine. Ah, still, be careful

if the dice tells you something between eighteen and 21, go left rigth away ! Otherwise you'll end up meeting the Queen of Hearts. She's completely nuts, she's never let you leave. Really, between 18 and 21 included, left as fast as yeh can ! Hey, know what ? If you ever get a 1, look on top of yer head, means the Rabbit is here ! Got it ? Remmber all that ? Y'see, th'nots so complicated.". Alice was head over heels with all those numbers, but she rolled the dice and went behind the White Rabbit. While she was fading away, the Gecko yelled "Ah, I forget ! Whenever y'don't know what teh do, just throw the dice again !".

Write a function named 'follow_the_white_rabbit'. This function must follow the journey of Alice. You will use random(3) to simulate the dice being rolled.

When Alice must go left, you will print on the standard output "left" followed by a newline. If she must go right, you will display "right" followed by a newline, if she must keep going straight ahead, you will display "ahead" followed by a newline, and if he must head back, you will display "back" followed by a newline ! Finally, when she finds the White Rabbit, you will display "RABBIT !!!" followed by a newline.
The function must return the sum of all the results the dice gave until now. You must provide one function only, do not provide a main function, the moulinette will take care of it. It will also take care of srandom(3), so you must NOT call it yourself.