ECE 70 – Computational Programming for Engineers
Test #2 – Professor Kriehn – Fall 2016
Tuesday, November 22, 2016

'D' Problem – Homework #32

Write a program that prompts the user for a random seed followed by a second prompt asking for the number of times a pair of dice are to be rolled. The program should generate the appropriate random numbers (between 1-6 for each die), sum the results, and store the sum in a variable length array. The output should print the number of times the sum of the two dice is odd or even, and the percentage of times the sum is odd or even.

Specifications:
Create a project called **`HW32`** in NetBeans and create a **`hw32.c`** source file. Make sure that the first line of **`hw32.c`** is **`/* homework 32 */`**.

Prompt the user to enter a random seed, using "`Enter a random seed: `" as the prompt. After reading in a seed value, use it seed the random number generator. Then prompt the user to enter the number of times a pair of dice are to be rolled, and store the result in `n`. Use "`Enter the number of times the dice are to be rolled: `" as the prompt. Then create a variable length array called `numbers[]` whose size is equal to `n`. Next call the `rand()` function to simulate rolling 2 dice (each producing a number between 1-6). Sum the result, and store each value in an element of the variable length array. Then determine how many times the sum of the two dice is odd or even, and the percentage of times the sum is odd or even. Print the results to the screen, and include one value after the decimal point for the percentages. Use the following variables:

> **`seed`** – stores random seed (unsigned integer)
> **`i`** – counter variable (integer)
> **`n`** – stores the number of elements of the array (integer)
> **`odd`** – stores the number of times the sum of the two dice are odd (integer)
> **`even`** – stores the number of times the sum of the two dice are even (integer)
> **`numbers[]`** – name of the variable length array (integer)

NOTE: You must store the sum of each roll of the two dice into an element of the `numbers[]` array if you want credit for this problem.

If you execute the program with the following underlined inputs, the results will be:

```
~> hw32.o
Enter a random seed: 10
Enter number of random integers to generate: 100

Number of times sum is odd: 48
Number of times sum is even: 52

Percentage of times sum is odd: 48.0%
Percentage of times sum is even: 52.0%
~>
```

Use the back page for your I/O diagram and pseudocode development. When you have double checked your work, submit your program to the grader program.

'C' Problem – Homework #33

Highlight your code from **`HW32`**, and copy it using **`Ctrl-C`**. Close your project, and create a new project called **`HW33`**, with a source file called **`hw33.`**c. Press **`Ctrl-V`** to copy the old **`hw32.c`** code into **`hw33.c`**.

Edit the first line of **`hw33.c`** to be **`/* homework 33 */`**.

Specifications:
Modify your program so that the roll of the two dice (and their sum) is produced in a function called `toss()`. Also create a function called `num_odd()` to calculate the total number of times the sum of the roll of the two dice is odd, and a function called `num_even()` to calculate the total number of times the sum is even. Use the following function prototypes:

```
int toss(void);
int num_odd(int n, int num[n]);
int num_even(int n, int num[n]);
```

When called, the `toss()` function rolls two dice (each producing a number between 1-6) and returns the sum. The `num_odd()` function determines the number of times the sum is odd, based on the data in the array. The `num_even()` function determines the number of times the sum is even, based on the data in the array.

<u>NOTE: You must create and use the toss(), num_odd(), and num_even() Functions if you want credit for this problem.</u>

If you execute the program with the following underlined inputs, the results will be:

```
~> hw33.o
Enter a random seed: 10
Enter number of random integers to generate: 100

Number of times sum is odd: 48
Number of times sum is even: 52

Percentage of times sum is odd: 48.0%
Percentage of times sum is even: 52.0%
~>
```

Use the back page for your I/O diagram and pseudocode development. When you have double checked your work, submit your program to the grader program.

'B' Problem – Homework #34

Highlight your code from **`HW33`**, and copy it using **`Ctrl-C`**. Close your project, and create a new project called **`HW34`**, with a source file called **`hw34.`**c. Press **`Ctrl-V`** to copy the old **`hw33.c`** code into **`hw34.c`**.

Edit the first line of **`hw34.c`** to be **`/* homework 34 */`**.

Specifications:
Remove your `num_odd()`and `num_even()`functions and replace them with an `odd_even()` function. Use the following function prototype:

```
void odd_even(int n, int *num, int *odd, int *even);
```

The purpose of the function is to calculate the total number of times the sum of the two dice (stored in the elements of the array) are odd and even. When called, the `odd_even()` function accepts an integer `n` representing the number of elements in the array, an address of the start of the array (passed to the integer pointer `*num`), an address where the calculation of the total number times the sum is odd will be stored (paseed into the integer pointer `*odd`), and an address where the number of times the sum is even will be stored (passed to the integer pointer `*even`).

Here, the integer pointer `*num` should point to the 0th element of the `numbers[]` array back in the main function, and be used to examine the data stored in the array. The integer pointer `*odd` should point to the `odd` variable back in the main function, and be used to store the number of times the sum of the two dice is odd. Likewise, the `*even` pointer should point to the `even` variable back in the main function, and be used to store the number of times the sum is even. Finally, create three additional pointers in the main function:

`*o` – integer pointer that points to the `odd` variable (<u>integer pointer</u>)
`*e` – integer pointer that points to the `even` variable (<u>integer pointer</u>)
`*num` – integer pointer that points to the `numbers[]` array (<u>integer pointer</u>)

Use these pointers to access the `odd` and `even` variables, and the `numbers[]` array, as appropriate.

<u>NOTE: You must create and use the `odd_even()`function,</u>
<u>and use pointers throughout your code if you want credit for this problem.</u>

If you execute the program with the following underlined inputs, the results will be:

```
~> hw34.o
Enter a random seed: 10
Enter number of random integers to generate: 100

Number of times sum is odd: 48
Number of times sum is even: 52

Percentage of times sum is odd: 48.0%
Percentage of times sum is even: 52.0%
~>
```

Use the back page for your I/O diagram and pseudocode development. When you have double checked your work, submit your program to the grader program.

'A' Problem – Homework #35

Highlight your code from **`HW34`**, and copy it using **`Ctrl-C`**. Close your project, and create a new project called **`HW35`**, with a source file called **`hw35.`**c. Press **`Ctrl-V`** to copy the old **`hw34.c`** code into **`hw35.c`**.

Edit the first line of **`hw35.c`** to be **`/* homework 35 */`**.

Specifications:
Modify your program to print out the sum of the two dice each time the dice are rolled. To do this, create an array of character pointers `*p[]` that contains the following strings:

“Two”	“Three”	“Four”	“Five”
“Five”	“Six”	“Seven”	“Eight”
“Nine”	“Ten”	“Eleven”	“Twelve”

The ragged pointer array should be declared as a constant, global variable. Modify the main program so that after a random sum is generated, you use the ragged array to print the word of the sum that is generated.

NOTE: You must create and use a ragged array using an array of character pointers if you want credit for this problem.

If you execute the program with the following underlined inputs, the results will be:

```
~> hw35.o
Enter a random seed: 10
Enter the number of times the dice are to be rolled: 10

The sum is: Seven
The sum is: Nine
The sum is: Seven
The sum is: Eight
The sum is: Twelve
The sum is: Twelve
The sum is: Five
The sum is: Five
The sum is: Six
The sum is: Seven

Number of times sum is odd: 6
Number of times sum is even: 4

Percentage of times sum is odd: 60.0%
Percentage of times sum is even: 40.0%
~>
```

Use the back page for your I/O diagram and pseudocode development. When you have double checked your work, submit your program to the grader program.