Assignment: create a State Lottery Program.

50 points:

Write a program which behaves similarly to a State Lottery.

When run, the program will produce six unique random

numbers between 1 and 50, inclusive.

EXAMPLE RUNS:

a.out

34 19 04 31 22 06

a.out

04 11 25 34 37 47

The output should be "random enough", i.e., pseudorandom,

non-repeating from run to run.

This is more an exercise in imagination and design, the programming is

simple. You must prevent duplicate numbers in each run, so figure out

how to do that. (The program must remember what numbers it's already

generated, and make sure it doesn't produce any duplicates.) Use your

imagination to come up with a way for the program to know what numbers

it's already printed, so it doesn't repeat.

Juice:

There are at least three distinct ways to do this. One using an array

of 50 numbers, one using an array of 6 numbers, and one brain-damaged

way that doesn't use arrays at all.

Whichever you use, BE SURE YOU UNDERSTAND IT.

Never turn anything in you can't explain.

Tools:

while (count < 6)

for( ; ; )

int arrayname[size], x, count, newnum

if ( == )

Pseudocode Ruminations:

(program start)

(loop)

generate a new random number

find out if this new number is unique

if it is, store it in the array, and

incriment count

if it is not, force the next iteration of the loop

(loop end: if count == 6, quit the loop, else run it again)

loop setting x for 0 to 5

print out the contents of array[x]

(program end)

All contents copyright (C) T.E.H., all rights reserved. None of this

text may be displayed or stored on any public access or private access

system without the written consent of the author.

/\*-----------------------------------------------------------------

\* powerball0.cpp - produce six unique random lottery numbers

Begun: Mon Sep 6 21:57:52 PDT 2010

Rank Beginner: Simplistic.

Descrete variables, no array.

Change MOD to 50 for prod.

-----------------------------------------------------------------\*/

#include <stdio.h>

#include <stdlib.h> //rand, srand

#include <time.h>

#define MOD 6

int main ()

{

/\*

this logic is DAIN BRAMAGED

\*/

int first, second, third, fourth, fifth, sixth ;

srand(time(0)) ;

first = rand() % MOD + 1 ;

two: second = rand() % MOD + 1 ;

if (second == first) goto two ;

three: third = rand() % MOD + 1 ;

if (third == first || third == second ) goto three ;

four: fourth = rand() % MOD + 1 ;

if (fourth == first || fourth == second || fourth == third ) goto four ;

five: fifth = rand() % MOD + 1 ;

if (fifth == first || fifth == second

|| fifth == third || fifth == fourth ) goto four ;

six: sixth = rand() % MOD + 1 ;

if (sixth == first || sixth == second || sixth == third

|| sixth == fourth || sixth == fifth) goto four ;

printf("%2d %2d %2d %2d %2d %2d\n",

first, second, third, fourth, fifth, sixth) ;

} // main

/\*-----------------------------------------------------------------

\* powerball1.cpp - produce six unique random lottery numbers

Begun: Mon Sep 6 21:57:52 PDT 2010

The no-dupes logic creates an array of 50 (51, but we aren't using 0)

numbers. When a number is generated, THAT element in the array

gets a non-zero value.

Intermediate: overly complex, but it does show recalc invocation.

Change MOD to 50 for prod.

-----------------------------------------------------------------\*/

#include <stdio.h>

#include <stdlib.h> //rand, srand

#include <string.h> //memset

#define MOD 6

int main ()

{

/\*

slightly easier to maintain, since it uses an array,

but a very wasteful and slow-processing one.

\*/

static int domain[MOD+1] ;

unsigned int s ;

srand(time(0)) ;

memset (domain, 0 , sizeof(int) \* MOD+1) ; // set to 0s

int i, n ;

for (i = 0 ; i < 6 ; ++i)

{

RAN:

n = (rand() % MOD ) + 1 ;

if (domain[n]) // if non-zero, we've already got that one

{

putchar('.') ; // track re-comps

goto RAN ;

}

domain[n] = n ;

printf("%02d ", n) ;

} // for i

putchar('\n') ;

// for ( i = 1 ; i < MOD+1 ; ++i) if (domain[i]) printf("%02d ", i) ;

putchar('\n') ;

} // main

/\*-----------------------------------------------------------------

\* powerball2.cpp - produce six unique random lottery numbers

Begun: Mon Sep 6 21:57:52 PDT 2010

The no-dupes logic uses a 6-element array, and for every

number generated, it compares all previous numbers. If the

current number is a dupe, the logic repeats until it isn't.

Change MOD to 50 for prod.

-----------------------------------------------------------------\*/

#include <stdio.h>

#include <stdlib.h> //rand, srand

#include <string.h> //memset

#define MOD 6

int main (int argc, char \*argv[], char \*\*env)

{

/\*

I consciously use goto to process error conditions, since

so many modern textbooks omit the command, yet it is

how computer hardare impliments loops and conditionals.

disapproving things as policy is superstition.

Look up Bruce Lee's comments to James Franciscus

on "Longstreet" regarding biting when fighting.

goto is just about the same: use it when necessary,

don't make a plan of it.

\*/

int domain[6] = {0} ;

unsigned int s ;

srand(time(0)) ;

memset (domain, 0 , sizeof(int) \* 6) ;

int i, n, j ;

for (i = 0 ; i < 6 ; ++i) {

RAN:

n = (rand() % MOD ) + 1 ;

for (j = 0 ; j <= i ; ++j) { // check for duplicates

if (domain[j] == n) { // duplicate?

putchar('.') ;

goto RAN ;

}

}

printf("%02d ", domain[i] = n) ;

} // for i

putchar('\n') ;

} // main

/\*-----------------------------------------------------------------

\* powerball2.cpp - produce six unique random lottery numbers

Begun: Mon Sep 6 21:57:52 PDT 2010

The PSEUDO RANDOM NUMBER GENERATOR SEED here uses my "type something"

code. This will insure that the run is as random as an average

programmer can manage and still run on Windows, which doesn't

have Linux' /dev/random and /dev/urandom entropy taps.

Change MOD to 50 for production code.

-----------------------------------------------------------------\*/

#include <stdio.h>

#include <stdlib.h> //rand, srand

#include <string.h> //memset

#include <ctype.h> //memset

#include <time.h>

#include <sys/time.h>

#define MOD 6

unsigned long keyseed(void) ;

int main ()

{

/\*

note that I'm allowing a visual cue so the user can see

how many times the code has to recalculate due to dupes.

by setting MOD to 6, you are in a better position to

test the duplicate detection. Be sure you understand

the point of this.

\*/

int domain[6] = {0} ;

unsigned int s ;

clock\_t seed = (keyseed()) ;

//printf("seed: %u\n", seed ) ;

srand(seed) ;

memset (domain, 0 , sizeof(int) \* 6) ;

int i, n, j ;

for (i = 0 ; i < 6 ; ++i) {

RAN:

n = (rand() % MOD ) + 1 ;

for (j = 0 ; j <= i ; ++j) { // check for duplicates

if (domain[j] == n) { // duplicate?

putchar('.') ;

goto RAN ;

}

}

printf("%02d ", domain[i] = n) ;

} // for i

putchar('\n') ;

} // main

/\*-----------------------------------------------------------------

\* randomseed2.cpp - seed the randomizer by measuring time b/t

\* keystrokes as user types.

Thu Jul 9 16:18:50 PDT 2009

-----------------------------------------------------------------\*/

#include <stdio.h>

#include <ctype.h>

#include "/var2/local/include/getchne.h"

/\*---------------

struct timeval {

time\_t tv\_sec; seconds

unsigned long tv\_usec; microseconds //suseconds\_t tv\_usec; microseconds

};

---------------\*/

unsigned long keyseed(void)

{

struct timeval timeval ;

#define REPS 14

#ifdef DEBUG

printf("sizeof(clock\_t\_t: %d\n", sizeof(clock\_t)) ;

#endif

clock\_t times[REPS] , start, stop ;

clock\_t seed = 0 ;

int i, count = 0 ;

printf("type something") ;

do

{

gettimeofday(&timeval,NULL) ;

start = timeval.tv\_usec ; // microseconds

getchne() ; // getch() in Visual C++

putchar('\b') ; putchar(' ') ; putchar('\b') ; // erase a char

gettimeofday(&timeval,NULL) ;

stop = timeval.tv\_usec ;

#ifdef DEBUG

printf("stop: %u start: %u\n", stop, start) ;

#endif

times[count] = stop - start ; // store in array

count++ ;

} while (count < REPS) ;

for (i = 0 ; i < REPS ; i++)

{

seed ^= times[i] ; // combine using XOR to get a seed

}

#ifdef DEBUG

printf("seed: %u\n", seed ) ;

#endif

return seed ;

} //keyseed ends