**Random Number Generator**

**Writer:** Sayak Haldar

**1. The function srand(unsigned)**a) Sets the seed for rand  
b) Doesn’t exist  
c) Is an error  
d) None of the mentioned  
  
**Answer is a)** Sets the seed for rand

1. **Which is the best way to generate numbers between 0 to 99?**a) rand()-100  
   b) rand()%100  
   c) rand(100)  
   d) srand(100)  
     
   **Answer is b) rand()%100**  
   **3. The correct way to generate numbers between minimum and maximum(inclusive) is \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**a) minimum + (rand() % (maximum – minimum));  
   b) minimum + (rand() % (maximum – minimum + 1));  
   c) minimum \* (rand() % (maximum – minimum))  
   d) minimum – (rand() % (maximum+minimum));  
     
   **Answer is b) minimum+rand()%(maximum-minimum+1)**  
     
   **4. rand() and srand() functions are used**  
   a) To find sqrt  
   b) For and operations  
   c) For or operations  
   d) To generate random numbers  
     
   **Answer) d) to generate random numbers.**   
     
   **5. What is the return type of rand() function?**  
   a) short  
   b) int  
   c) long  
   d) double
2. **Answer is b) int**  
     
   **6. Which of the following can be used for random number generation?**a) random()  
   b) rnd()  
   c) rndm()  
   d) none of the mentioned  
     
     
     
   **7. Which of the following snippet will effectively generate random numbers?**  
   a) rand();  
   b) rand(10);  
   c) rand(time(NULL));  
   d) all of the mentioned  
     
   **Answer is a) rand()**

**Explanation:** rand() generates a random number between 0 to RAND\_MAX

**8. Which among the following is correct function call for rand and random?**a) rand() and random();  
b) rand() and random(1);  
c) rand(1) and random(1);  
d) rand(1) and random();  
  
**Answer) is a)**  
  
**9. For the function call time(), what type of parameter is accepted?**a) int  
b) int \*  
c) time\_t  
d) time\_t \*

**Answer is d) time\_t \***

**Explanation:** (this is needed before srand())

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main()

{

int i, n;

time\_t t;

n = 5;

/\* Initializes random number generator \*/

srand((unsigned) time(&t));

//So, check here. We are sending the address of the time\_t variable to time function //Hence, we can say, it accepts the argument of time\_t \* /\* Print 5 random numbers from 0 to 50 \*/

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

{

printf("%d\n", rand() % 50);

}

return(0);

}

1. **What is the output of this C code?**

#include <stdio.h>

#include <stdlib.h>

int main()

{

printf("%d\n", rand() % 1000);

return 0;

}

a) Compile time error

b) An integer between 0-1000

c) An integer between 0-999 including 0 and 999

d) An integer between 0-1000 including 1000  
  
**Answer) c)**

1. **What is the output of this C code?**

#include <stdio.h>

#include <stdlib.h>

int main()

{

srand(9000);

printf("%d\n", rand());

return 0;

}

a) Compile time error

b) An integer in the range 0 to RAND\_MAX

c) A double in the range 0 to 1

d) A float in the range 0 to 1

**Answer is b)An integer in the range 0 to RAND\_MAX.**

1. **What is the output of this C code?**

#include <stdio.h>

int main()

{

printf("%d\n", srand(9000));

return 0;

}

a) Compile time error

b) An integer in the range 0 to 9000

c) A float in the range 0 to 1

d) A double in the range 0 to 9000

**Answer is a) Compile time error**

**Explanation:**

Compiler would say: invalid use of void expression in printf("%d\n", srand(9000)); Remember, return type of srand() is void.

1. **What is the output of this C code?**

#include <stdio.h>

int main()

{

srand(time(NULL));

printf("%d\n", rand());

return 0;

}

a) Compile time error

b) An integer in the range 0 to RAND\_MAX

c) A double in the range 0 to 1

d) A float in the range 0 to 1

**Answer is b) An integer in the range 0 to RAND\_MAX**

1. **In the below program everytime program is run different numbers are generated.**

#include <stdio.h>

#include <stdlib.h>

int main()

{

printf("%d\n", rand());

return 0;

}

a) True

b) False

c) Depends on the platform

d) Depends on the compiler

**Answer is b) false.**

**Explanation:** It will generate same number every time. To generate different random number every time, we have to use srand() function.

We can type the previous function to get different random number every time.

In the below program everytime program is run different numbers are generated.

1.#include <stdio.h>

2.int main()

3.{

4. srand(time(NULL));

5. printf("%d\n", rand());

6. return 0;

7.}

**The srand() function sets its argument as the seed for a new sequence of pseudo-random integers to be returned by rand(). These sequences are repeatable by calling srand() with the same seed value.**

Now, what does that mean?

If no seed value is provided, the **rand**() function is automatically seeded with a value of 1.

Now, some conceptual question:

1.#include <stdio.h>

2.int main()

3.{

4. srand(time(NULL));

5. printf("%d\n", rand());

6. return 0;

7.}

Why this function generates a different number every time the number is defined?

Because, of srand(time(null))

makes use of the computer's internal clock to control the choice of the seed. Since time is continually changing, the seed is forever changing. Remember, if the seed number remains the same, the sequence of numbers will be repeated for each run of the program.

**Now, How do rand() and srand() implementations actually work?**

**Sample of rand() and srand() implementations:**

POSIX.1-2001 gives the following example of an implementation of **rand**() and **srand**(), possibly useful when one needs the same sequence on two different machines.

static unsigned long next = 1;

/\* RAND\_MAX assumed to be 32767 \*/

int myrand(void) {

next = next \* 1103515245 + 12345;

return((unsigned)(next/65536) % 32768);

}

void mysrand(unsigned seed)

{

next = seed;

}

1. **In the below program everytime program is run different numbers are generated.**

#include <stdio.h>

int main()

{

srand(time(NULL));

printf("%d\n", rand());

return 0;

}

a) True

b) False

c) Depends on the platform

d) Depends on the compiler

**Answer) a) true.**

1. **Which of these is a correct way to generate numbers between 0 to 1(inclusive) randomly?**

a) rand() / RAND\_MAX

b) rand() % 2

c) rand(0, 1)

d) none of the mentioned

**Answer is b) rand()%2**

**Extras: (From GeeksForGeeks And StackOverFlow)  
  
rand ()**

rand() function is used in C to generate random numbers. If we generate a sequence of random number with rand() function, it will create the same sequence again and again every time program runs. Say if we are generating 5 random numbers in C with the help of rand() in a loop, then every time we compile and run the program our output must be the same sequence of numbers.

**Syntax:**

**int rand(void):**

returns a pseudo-random number in the range of 0 to RAND\_MAX.

**RAND\_MAX:** is a constant whose default value may vary

between implementations but it is granted to be at least 32767.

**// C program to generate random numbers**

**#include <stdio.h>**

**#include <stdlib.h>**

**// Driver program**

**int main(void)**

**{**

**// This program will create same sequence of**

**// random numbers on every program run**

**for(int i = 0; i<5; i++)**

**printf(" %d ", rand());**

**return 0;**

**}**

**NOTE:** This program will create same sequence of random numbers on every program run.

**Output 1:**

453 1276 3425 89

**Output 2:**

453 1276 3425 89

**Output n:**

453 1276 3425 89

**srand()**

The srand() function sets the starting point for producing a series of pseudo-random integers. If srand() is not called, the rand() seed is set as if srand(1) were called at program start. Any other value for seed sets the generator to a different starting point.

**Syntax:**

**void srand( unsigned seed ):**

Seeds the pseudo-random number generator used by rand()

with the value seed.

**Note:** The pseudo-random number generator should only be seeded once, before any calls to rand(), and the start of the program. It should not be repeatedly seeded, or reseeded every time you wish to generate a new batch of pseudo-random numbers.

Standard practice is to use the result of a call to srand(time(0)) as the seed. However, time() returns a time\_t value which vary everytime and hence the pseudo-random number vary for every program call.

**// C program to generate random numbers**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include<time.h>**

**// Driver program**

**int main(void)**

**{**

**// This program will create different sequence of**

**// random numbers on every program run**

**// Use current time as seed for random generator**

**srand(time(0));**

**for(int i = 0; i<5; i++)**

**printf(" %d ", rand());**

**return 0;**

**}**

NOTE: This program will create different sequence of random numbers on every program run.

**Output 1:**

453 1432 325 89

**Output 2:**

8976 21234 45 8975

**Output n:**

563 9873 12321 24132

**How srand() and rand() are related to each other?**

srand() sets the seed which is used by rand to generate “random” numbers. If you don’t call srand before your first call to rand, it’s as if you had called srand(1) to set the seed to one.

In short, srand() — Set Seed for rand() Function.

**What Does rand\_r function do?**  
 rand\_r() returns a pseudo-random integer in the range [0, RAND\_MAX]. The seedp argument is a pointer to an unsigned int that is used to store state between calls. If rand\_r() is called with the same initial value for the integer pointed to by seedp, and that value is not modified between calls, then the same pseudo-random sequence will result.

he value pointed to by the seedp argument of rand\_r() provides only a very small amount of state, so this function will be a weak pseudo-random generator. Try drand48\_r() instead.

That is the basic.

**But, what is the application of these functions? Why rand\_r() is better than drand48\_r()?**you need not to know this.

**A Typical Question Using rand\_r()**

**What should we do, if in a multithreading environment we need different seed value for different thread?**

For the reentrant version rand\_r, the seed is just the initial value of the state .You need one seed per thread. Either create an array of seeds, or make the seed variable thread-local:

**\_Thread\_local unsigned int seed = time(NULL);**

**int do\_stuff()**

**{**

**for ( ; ; )**

**{**

**int n = rand\_r(&seed);**

**// use n**

**}**

**}**