The `rand()` function is part of the stdlib. When called it returns a pseudorandom integer in the range 0 to `RAND_MAX` where `RAND_MAX` is a constant defined in `stdlib.h`. For Windows machines running Bloodshed or Visual C++, `RAND_MAX` has a value of $32,757 = 2^{15} - 1$. Consider the following program which uses the `rand()` function to print the value of `RAND_MAX` and 10 random integers.

```cpp
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    int i;
    cout << RAND_MAX << endl;
    for(i=0; i<10; i++)
        cout << rand() << "  ";
    cout << "Push return to continue...";
    cin.get();
    return 0;
}
```

The output from this program is:

```
32767
41 18467 6334 26506 19169 15724 11478 29358 26962 24464
Push return to continue...
```

Note that if you run this program multiple times on different computers you will always get the same output so one might ask the question "How random are these random numbers if they can be repeated?" The answer is that the numbers produced are not random at all but they have the appearance of being random. They are called pseudorandom numbers because they are created by a repeatable deterministic process. Pseudorandom numbers have the mathematical characteristics of randomness and can therefore be used to substitute for truly random numbers in some circumstances. For example you might want to use the rand function to generate random integers from 1 to 6 to simulate the roll of a die. You can do this by modifying the use of the rand function as shown below.

```cpp
int die;
die = (rand() % 6) + 1;
```

In this set of statements the `%` is the mod operator, i.e. it returns the remainder after division. So, for example, if the rand function produced the number 41, the operation `rand() % 6` would produce the number 5 since $41/6 = 6$ remainder 5. Since division by 6 always produces a number from 9 to 5 as a remainder we add one to get a random number 1 to 6. In general, to produce a random integers from rLow to rHigh use the following equation:
random = (rand() %(rHigh-rLow+1)) + rLow;

One other potential problem with the rand function is that the starting sequence appears to be the same so that if you created a game using rand a smart player could predict the outcome after trying your game a few times because the numbers generated would always be the same. In C++ there is another function that allows you to fix this. This function is called srand() and it allows you to seed the random number generator with a new starting point. Typically you run srand one time at the beginning of your program to give the rand function an arbitrary starting point. To get a seed that changes every time you use it you can use the time function which gives the number of seconds since midnight. The sequence below produces a different set of random numbers each time it executes since the seed value changes with the time.

```cpp
#include <iostream>
#include <cstdlib>            //needed for rand and srand functions
#include <ctime>              //needed for time function
using namespace std;

int main()
{int i;
 srand(time(0));
 cout << RAND_MAX << endl;
 for(i=0;i<10;i++)
   cout << rand() << " ";
 cout << "\nPUSH return to continue...";
 cin.get();
 return 0;
}
```