Lab Report: Rolling Dice Generator & Random Password Generator

Objective

- To simulate a Rolling Dice Generator program using C.
- To simulate the Random Password Generator program using C.

Theory

1. Rolling Dice Generator:

This program simulates rolling a six-sided die using random number generation. The 'rand()' function in the C Standard Library is used to generate numbers between 1 and 6, representing the faces of a die. The program counts the occurrences of each face across multiple rolls and calculates the probabilities by dividing the count of each face by the total number of rolls. The randomness is seeded with the current system time using 'srand(time(NULL))' to ensure that each execution produces different results.

2. Random Password Generator:

This program generates a random password of a specified length by selecting characters from a predefined set, which includes uppercase letters, lowercase letters, numbers, and special characters. The random selection is achieved using the 'rand()' function, and the randomness is seeded similarly with the current system time. The password is stored in a character array and null-terminated to form a valid C string. The use of diverse character types ensures strong and secure passwords.

Observation:

1. Random Password Generator

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void generatePassword(int length)
```

```
{
   char regex[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
                  "abcdefghijklmnopqrstuvwxyz"
                  "0123456789"
                  "!@#$%^&*()-_=+[]{};:,.<>?";
int regex_size = sizeof(regex) - 1;     char
password[length + 1];     for (int i = 0; i <
length; i++)
   {
      int randomIndex = rand() % regex_size;
password[i] = regex[randomIndex];
   }
   password[length] = '\0'; // null-terminate string
printf("Generated Password: %s\n", password);
} int main(){     int length;
srand(time(NULL));     printf("Enter the
desired password length: ");     scanf("%d",
&length);     generatePassword(length);
return 0;
}
```

Output:

Enter the desired password length: 10

Generated Password: =CpAO%IHY7

Discussion:

The **Random Password Generator** program uses rand() with a diverse character set (uppercase, lowercase, digits, special characters) to generate secure and unpredictable passwords. The password length is user-defined, and the randomness is seeded with srand(time(NULL)) for varied outputs. The implementation ensures strong, unique passwords resistant to brute force attacks.

2. Rolling dice generator

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {     int n,
result;     int counts[6]
= {0};
srand(time(NULL));

   printf("Enter the number of dice rolls: ");
scanf("%d", &n);

   printf("Rolling a six-sided die %d times:\n", n);
for (int i = 0; i < n; i++)
   {         result = (rand() % 6)
+ 1;         counts[result - 1]++;
printf("Roll %d: %d\n", i + 1,
result);
   }

   // Calculate and display probabilities
printf("\nFace\tCount\tProbability\n");     for
(int i = 0; i < 6; i++)
   {
      double probability = (double)counts[i] / n;
printf("%d\t%d\t%.2f\n", i + 1, counts[i], probability);
   }

   return 0;
```

```
}
```

Output:

```
Enter the number of dice rolls: 10 Rolling
a six-sided die 10 times:
Roll 1: 1
Roll 2: 4
Roll 3: 4
Roll 4: 4
Roll 5: 1
Roll 6: 4
Roll 7: 1
Roll 8: 1
Roll 9: 6
Roll 10: 3

Face   Count   Probability
1      4       0.40
2      0       0.00
3      1       0.10
4      4       0.40
5      0       0.00
6      1       0.10
```

Conclusion:

Both programs successfully demonstrate the use of random number generation and string manipulation in C programming. The Rolling Dice Generator provides insights into probability calculations and statistical analysis, while the Random Password Generator highlights the practical application of randomization in creating secure passwords. These programs serve as foundational examples for developing tools in the domains of games and information security.