**1.Generating random number using rand() function**

Problem Statement:

(a) Create a function that randomly generates 10000 numbers from 0 to 100.

(b) Consider the random numbers generated in (a) are ages of 10000 people living in your locality/village/town. You need to count how many people are of a particular age, i.e., number of people with age 0, 1, 2, 3, ..., 100. Print this list as

Age a = count

Print number of people in each age groups [0 − 10], [11 − 20], · · · , [91 − 100]. See the execution time of your program.

(c) Print the bar plot of count of people in all the 10 age groups.

Use '\*' or any other special character to plot the graph.

* Output Example:

Number of people in every age:

Age 0 = 111

Age 1 = 87

Age 2 = 111

Age 3 = 115

Age 4 = 113

Age 5 = 100

Age 6 = 89

Age 7 = 102

Age 8 = 99

Age 9 = 95

Age 10 = 80

Age 11 = 92

Age 12 = 102

Age 13 = 82

Age 14 = 74

Age 15 = 92

Age 16 = 103

Age 17 = 96

Age 18 = 74

Age 19 = 89

Age 20 = 111

Age 21 = 89

Age 22 = 87

Age 23 = 89

Age 24 = 103

Age 25 = 88

Age 26 = 94

Age 27 = 98

Age 28 = 102

Age 29 = 99

Age 30 = 87

Age 31 = 90

Age 32 = 111

Age 33 = 100

Age 34 = 99

Age 35 = 94

Age 36 = 112

Age 37 = 111

Age 38 = 103

Age 39 = 79

Age 40 = 87

Age 41 = 90

Age 42 = 109

Age 43 = 110

Age 44 = 85

Age 45 = 78

Age 46 = 101

Age 47 = 96

Age 48 = 94

Age 49 = 116

Age 50 = 103

Age 51 = 98

Age 52 = 101

Age 53 = 104

Age 54 = 110

Age 55 = 88

Age 56 = 115

Age 57 = 121

Age 58 = 100

Age 59 = 91

Age 60 = 108

Age 61 = 103

Age 62 = 104

Age 63 = 96

Age 64 = 109

Age 65 = 118

Age 66 = 89

Age 67 = 110

Age 68 = 102

Age 69 = 103

Age 70 = 93

Age 71 = 101

Age 72 = 98

Age 73 = 107

Age 74 = 94

Age 75 = 115

Age 76 = 74

Age 77 = 105

Age 78 = 112

Age 79 = 95

Age 80 = 98

Age 81 = 99

Age 82 = 101

Age 83 = 103

Age 84 = 109

Age 85 = 88

Age 86 = 99

Age 87 = 101

Age 88 = 88

Age 89 = 107

Age 90 = 99

Age 91 = 113

Age 92 = 94

Age 93 = 75

Age 94 = 113

Age 95 = 95

Age 96 = 106

Age 97 = 96

Age 98 = 114

Age 99 = 107

Age 100 = 110

Number of people between age groups:

0 to 10 -> 1102

11 to 20 -> 915

21 to 30 -> 936

31 to 40 -> 986

41 to 50 -> 982

51 to 60 -> 1036

61 to 70 -> 1027

71 to 80 -> 999

81 to 90 -> 994

91 to 100 -> 1023

The people / age graph:

\*\*\*

\*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

\*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\* \*\*\*

0- 10 11- 20 21- 30 31- 40 41- 50 51- 60 61- 70 71- 80 81- 90 91-100

Total time taken by CPU: 0.300000

Proposed C Code:

/\* -------main.c ------- \*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

*/\*Function to generate 10000 random number\*/*

int \*generatenum(int n, int \*count)

{

    srand(time(0));

    int \*nums = (int \*)calloc(10000, sizeof(int));

    for (int i = 0; i < 10000; i++)

    {

        nums[i] = rand() % 101; *// to squeeze the numbers greater than 100*

        count[nums[i]]++; *// calculating how many people are in a particular age*

    }

    return nums;

}

int main()

{

    clock\_t start\_t, end\_t;

    double total\_t;

    start\_t = clock(); *// starting the clock*

    int n = 10000;

    int \*count = (int \*)calloc((101), sizeof(int));

    int \*arr = generatenum(n, count); *// generate 10000 numbers and store it in an array*

    printf("Number of people in every age:\n");

    for (int i = 0; i < 101; i++)

    {

        printf("Age %d = %d\n", i, count[i]);

    }

    printf("\n");

    int \*barcount = (int \*)calloc(10, sizeof(int));

    printf("Number of people between age groups:\n");

    for (int i = 1; i < 101; i += 10)

    {

        int c = 0;

        for (int j = i; j < i + 10; j++)

        {

            c += count[j]; *// counting how many people fall on each age groups*

        }

        barcount[i / 10] = c / 10; *// storing the numbers in an array*

        if (i == 1)

        {

            printf("%d to %d -> %d\n", 0, i + 9, c + count[0]);

        }

        else

        {

            printf("%d to %d -> %d\n", i, i + 9, c);

        }

    }

    printf("\n");

    int max = -1;

    for (int i = 0; i < 10; i++)

    {

        if (barcount[i] > max)

        {

            max = barcount[i]; *// finding the max element from the array*

        }

    }

*/\*Ploting the graph\*/*

    printf("The people / age graph: \n\n");

    for (int i = 0; i < max; i++)

    {

        for (int j = 0; j < 10; j++)

        {

            if (i < max - barcount[j])

            {

                printf("       ");

            }

            else

            {

                printf(" \*\*\*   ");

            }

        }

        printf("\n");

    }

    printf("0- 10 ");

    for (int i = 1; i < 10; i++)

    {

        printf("%d-%3d ", i \* 10 + 1, (i + 1) \* 10);

    }

    printf("\n");

    end\_t = clock(); *// ending the clock*

    total\_t = (double)(end\_t - start\_t) / CLOCKS\_PER\_SEC; *// counting the total time*

    printf("\nTotal time taken by CPU: %f\n", total\_t);

    return 0;

}

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

Conclusion:

The proposed algorithm has a runtime of O(n) where n is the input size under

consideration.

Limitations and assumptions for this algorithm include:

1. As the input size is 10000 so the frequency of each group is very large. So we are plotting the bar graph with 1:10 scale for clear visibility, due to this there are two very close values so the height of the bar columns may be almost same.
2. Here we are considering 0 to 100 age over 10000 people, so all the calculation are based on these assumptions.