Assignment 3

Task 1: Write statements to accomplish each of the following:

a) Display the value of the seventh element of character array f.

```
printf("%c", f[6]);
```

b) Input a value into element 4 of single-subscripted floating-point array b.

```
b[4] = 1.0;
```

c) Initialize each of the five elements of single-subscripted integer array g to 8.

```
int g[5] = {8, 8, 8, 8, 8};
```

d) Total the elements of floating-point array c of 100 elements.

```
float sum;
for (int x = 0; x < 100; x++)
{
    sum += c[x];
}
```

e) Copy array a into the first portion of array b. Assume double a[11], b[34];

```
for (int x = 0; x < 11; x++) b[x] = a[x];
```

f) Determine and print the smallest and largest values contained in 99-element floating- point array w.

```
float min = w[0];
float max = w[0];
for (int x = 0; x < 99; x++)
{
    if (w[x] < min) min = w[x];
    if (w[x] > max) max = w[x];
}
```

Task 1: Write a program that simulates coin tossing. For each toss of the coin the program should print Heads or Tails. Let the program toss the coin 100 times, and count the number of times each side of the coin appears. Print the results. The program should call a separate function flip that takes no arguments and returns 0

for tails and 1 for heads. [Note: If the program realistically simulates the coin tossing, then each side of the coin should appear approximately half the time for a total of approximately 50 heads and 50 tails.]

Paste your program code in the box below

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

int coinFlip()
{
    return rand() % 2;
}

int main()
{
    srand(time(NULL));

    int heads = 0;
    int tails = 0;

    for (int x = 0; x < 100; x++)
    {
        if (coinFlip()) heads++;
        else tails++;
    }

    printf("Heads: %d\nTails: %d", heads, tails);
    return 0;
}
```

Paste your program output in the box below

```
❯ tcc -run .\assingment3_2.c
Heads: 46
Tails: 54
```

Task 3: Write a function that takes three arguments: a character and two integers. The character is to be printed. The first integer specifies the number of times that

the character is to be printed on a line, and the second integer specifies the number of lines that are to be printed. Write a program that makes use of this function.

Paste your program code in the box below

```
#include <stdio.h>

void printer(char character, int times, int lines)
{
    for (int x = 0; x < lines; x++)
    {
        for (int y = 0; y < times; y++)
        {
            printf("%c", character);
        };
        printf("\n");
    };
}

int main()
{
    char character = '\0';
    int times = 0;
    int lines = 0;
    printf("Enter a character: ");
    scanf("%c", &character);
    printf("Enter how many times: ");
    scanf("%d", &times);
    printf("Enter how many lines: ");
    scanf("%d", &lines);
    printer(character, times, lines);
}
```

Paste your program output in the box below

```
❯ tcc -run .\assingment3_3.c
Enter a character: *
Enter how many times: 2
Enter how many lines: 2
**
```

```
**
```

Task 4: Write a function that returns the index of the largest value stored in an array-of-double

Paste your code in the box below

```
int largest(double data[], size_t length)
{
    double max = data[0];
    int max_index = 0;
    for (int x = 0; x < length; x++)
    {
        if (data[x] > max)
        {
            max = data[x];
            max_index = x;
        }
    };
    return max_index;
}
```

Task 5: Write a function that returns the difference between the largest and smallest elements of an array-of-int.

Paste your code in the box below

```
int difference(int data[], size_t length)
{
    int max = data[0];
    int min = data[0];

    for (int x = 0; x < length; x++)
    {
        if (data[x] > max)
        {
            max = data[x];
        }
        if (data[x] < min)
```

```
        {
            min = data[x];
        }
    };
    return max - min;
}
```

Task 6: Write a program that simulates the rolling of two dice.
The program should use rand() to roll the first die, and should use rand() again to roll the second die. The sum of the two values should then be calculated.

[Note: Since each die can show an integer value from 1 to 6, then the sum of the two values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 the least frequent sums.]

	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

The above table shows the 36 possible combinations of the two dice.
Your program should roll the two dice 36,000 times. Use a single-subscripted array to tally the numbers of times each possible sum appears. Print the results in a tabular format. Also, determine if the totals are reasonable, i.e., there are six ways to roll a 7, so approximately one-sixth of all the rolls should be 7.

Paste your program code in the box below

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

int diceRoll()
{
    return rand() % 6 + 1;
}
```

```
int diceRoller(int times)
{
    int sum = 0;
    for (int x = 0; x < times; x++)
    {
        sum += diceRoll();
    }
    return sum;
}

int main()
{
    srand(time(NULL));

    int tally[12] = {0};

    for (int x = 0; x < 36000; x++)
    {
        int hand = diceRoller(2);
        tally[hand - 2]++;
    }
    printf("Sum     |  Percentage of 36000 rolls\n");
    for (int y = 0; y < 11; y++)
    {
        printf("%d: %d      %f\n", y + 2, tally[y],
((float)tally[y] / 36000.0) * 100);
    }
    return 0;
}
```

Paste your program output in the box below

```
Sum     |  Percentage of 36000 rolls
2: 1054      2.927778
3: 1946      5.405556
4: 3031      8.419444
5: 4002      11.116667
```

```
6: 4962      13.783333
7: 5951      16.530556
8: 4916      13.655556
9: 4099      11.386111
10: 3042      8.450000
11: 2006      5.572222
12: 991      2.752778
```