Assignment 2

Task 1: Write a function *integerPower (base, exponent)* that returns the value of base ^{exponent}

For example, integerPower (3, 4) = 3 * 3 * 3 * 3. Assume that exponent is a positive, nonzero integer, and base is an integer. Function integerPower should use for to control the calculation. Do **NOT** use any math library functions.

Paste your program code in the box belowd

```
#include <stdio.h>

int integerPower(int base, int exponent)
{
    int sum = 1;
    for (; exponent > 0; exponent--)
    {
        sum = sum * base;
    }
    return sum;
}

int checkPositiveNonZero(int number)
{
    if (number < 0)
        return 1;
    if (number == 0)
        return 1;
    return 0;
}

int main()
{
    int base;
    int exponent;
    int error;

    printf("Enter an integer base: ");
```

```
    scanf("%d", &base);
    printf("Enter a positive integer exponent: ");
    scanf("%d", &exponent);

    error = checkPositiveNonZero(exponent);
    if (error)
    {
        printf("Invalid integer.");
        return error;
    }

    printf("%d", integerPower(base, exponent));
    return 0;
}
```

Task 2: An integer is said to be prime if it's divisible by only 1 and itself.
For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not.
Write a function that determines if an user input number is prime or not.

Paste your program code in the box below

```
#include <stdio.h>

int isPrime(int number)
{

    if ((number % 2 == 0) || (number % 3 == 0)|| (number %
5 == 0) || (number % 7 == 0)) return 1;
    return 0;
}

int main()
{
    int number;
    int result;

    printf("Enter an integer:");
    scanf("%d", &number);
```

```
    result = isPrime(number);

    if (result)
    {
        printf("%d is NOT a prime number", number);
        return 0;
    }
    printf("%d is a prime number", number);
    return 0;
}
```

Task 3: Write statements that assign random integers to the variable n in the following ranges:

a) $1 \leq n \leq 100$

```
    int n = (rand() % 100) + 1;
```

b) $-1 \leq n \leq 1$

```
    int n = (rand() % 3) - 1;
```

c) $-3 \leq n \leq 11$

```
    int n = (rand() % 15) - 3;
```

For each of the following sets of integers, write a single statement that will print a number at random from the set.

a) 2,4,6,8,10.

```
    printf("%d", set[rand() % 5])
```

b) 3,5,7,9,11.

```
    printf("%d", set[rand() % 5])
```

c) 6, 10, 14, 18, 22.

```
    printf("%d", set[rand() % 5])
```

Task 4: Write a function that displays a solid square of asterisks whose side is specified in integer parameter side. For example, if side is 4, the function displays:

```
****
****
****
****
```

Paste your program code in the box below

```
#include <stdio.h>

void printSquare(int side) {
    for (int x = 0; x < side; x++) {
        for (int y = 0; y < side; y++) {
            printf("* ");
        }
        printf("\n");
    }
}

int main() {
    int side;
    printf("Enter the length of the square: ");
    scanf("%d", &side);
    printSquare(side);
    return 0;
}
```

Task 5: We have triangle made of blocks. The topmost row has 1 block, the next row down has 2 blocks, the next row has 3 blocks, and so on. Write a recursive function (no loops or multiplication) returns the total number of blocks in such a triangle with the given number of rows.
Examples:

triangle(0) → 0
triangle(1) → 1
triangle(2) → 3

Paste your program code in the box below

```
#include <stdio.h>

int triangle(int rows) {
    if (rows <= 0) {
        return 0;
    } else {
        return rows + triangle(rows - 1);
    }
}

int main() {
    int rows;
    printf("Rows: ");
    scanf("%d", &rows);
    printf("Blocks: %d\n", triangle(rows));
    return 0;
}
```