coms527-hw1

Shengwei Mao

January 2023

1 Task 1

```
#include <stdio.h>

int main() {
    printf("Hello, world\n");
    return 0;
}
```

2 Task 2

```
/*
   The correction of the program will be in annotation

*/

#include <stdlib.h>

typedef struct
{
    char* street;
    int number;
    int post_code;
    char* city;
} address;

address* create_address (char* street, int number, int post_code, char* city)
{
    // In order to create a struct, we need to allocate memory for the struct
    address* new_address = (address *)malloc(sizeof(address));
    new_address->street = street;
    new_address->number = number;
```

```
    new_address->post_code = post_code;
    new_address->city = city;
    return new_address;
}

/*
    To transfer the data properly, the best way is to deliver
    pointer of the struct instead of struct itself.
*/
address* duplicate_address(address *orig)
{
    // new struct need memory, allocate memory first
    address* new_address = (address *)malloc(sizeof(address));
    /* To duplicate new struct, we need to copy the value
       of the original one to new memory. Using something
       like "new_address = orig" can only duplicate pointer,
       the data on the memory do not duplicate.
    */
    new_address->street = orig->street;
    new_address->number = orig->number;
    new_address->post_code = orig->post_code;
    new_address->city = orig->city;
    return new_address;
}

int main()
{
    address* a1 = create_address("Mornewegstr.", 30, 64293, "Darmstadt");

    // a1 is the address pointer, "&a1" is type "address**". We need to dele
    address* a2 = duplicate_address(a1);

    free(a1);
    free(a2);
}
```

3 Task 3

```
#include <stdio.h>

int main()
{
    for (int i = 1; i <= 5; i++) {
        for (int j = 0; j < i; j++) {
```

```
            printf("%d", j + 1);
        }
        printf("\n");
    }
    return 0;
}
```

4 Task 4

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

int isArmstrongs(int number)
{
    int orig = number;
    int degree = log10(number) + 1;
    int check = 0;
    while(number != 0){
        check += pow(number % 10, degree);
        number = number / 10;
    }
    return (orig == check);
}

int main() {
    printf("The following are Armstrong number within 10000000:\n");
    for(int i = 0; i < 10000000; i++){
        if(isArmstrongs(i) == 1){
            printf("%d\n", i);
        }
    }
    return 0;
}
```

5 Task 5

```
#include <stdio.h>
int main()
{
    for (int i = 5; i > 0; i--){
        for (int j = 1; j <= i; j++) {
            printf("%d", j);
        }
```

```
        printf("\n");
    }
    return 0;
}
```

6 Task 6

```
#include <stdio.h>
int Factorial(int n){
    if(n == 0) return 1;
    return n * Factorial(n-1);
}
int combination(int n, int m){
    return Factorial(n) / (Factorial(m) * Factorial(n-m));
}
void print_pascals_triangle(int row)
{
    for(int i = 0; i <= row; i++) {
        for(int j = 0; j <= i; j++) printf("%d ", combination(i, j));
        printf("\n");
    }
}
int main() {
    printf("Input the row needed for printing\n");
    int row = 0;
    scanf("%d", &row);
    printf("The following is the triangle you need\n");
    print_pascals_triangle(row);
    return 0;
}
```