1. Implement the following program by pointer to function

```c
int main(void)
{
    int a = 7, b = 3;

    printf("%d\n", calculate(a, b, '-'));
    printf("%d\n", calculate(a, b, '/'));
    printf("%d\n", calculate(a, b, '%'));

    return 0;
}
```

Implementation:

```c
#include<stdio.h>

int divide(int num1, int num2) {
    return num1 / num2;
}
int subtract(int num1, int num2) {
    return num1 - num2;
}

int modulo(int num1, int num2) {
    return num1 % num2;
}

typedef int (*fptrOperation)(int,int);


fptrOperation select(char opcode) {
    switch(opcode) {
    case '/': return divide;
    case '-': return subtract;
    case '%': return modulo;
    default : return NULL;
    }
}

```

```c
int calculate(int num1, int num2, char opcode) {
    fptrOperation operation = select(opcode);
    return operation(num1, num2);
}

int main(void){


    int a = 7,b=3;
    printf("%d\n", calculate(a, b, '-'));
    printf("%d\n", calculate(a, b, '/'));
    printf("%d\n", calculate(a, b, '%'));
    return 0;

}
```

2. Implement the Jump Tables on page 360 of the book Pointers-On-C.pdf

```c
switch( oper ){
case ADD:
      result = add( op1, op2 );
      break;

case SUB:
      result = sub( op1, op2 );
      break;

case MUL:
      result = mul( op1, op2 );
      break;

case DIV:
      result = div( op1, op2 );
      break;
```

Implementation:

```c
#include <stdio.h>


// Function prototypes
double add(double a, double b) { return a + b; }
double sub(double a, double b) { return a - b; }
double mul(double a, double b) { return a * b; }
double div(double a, double b) { return b != 0 ? a / b : 0; }

// Define function pointer type
typedef double (*operation)(double, double);

// Create jump table
operation operations[] = {
    add, sub, mul, div
};


```

```c

int main() {
    double op1, op2, result;
    int oper;

    // Get input from user
    printf("Enter first number: ");
    scanf("%lf", &op1);
    printf("Enter second number: ");
    scanf("%lf", &op2);
    printf("Enter operation (0:ADD, 1:SUB, 2:MUL, 3:DIV): ");
    scanf("%d", &oper);

    // Validate operation
    if (oper >= 0 && oper <= 3) {
        // Use jump table to perform operation
        result = operations[oper](op1, op2);
        printf("Result: %.2f\n", result);
    } else {
        printf("Invalid operation\n");
    }

    return 0;
}
```

3. The main function and output have already been provided, to complete the remaining parts.

```c
void main(void)
{
    int int_a = 10;
    char str1[] = "bdcom";
    person_t person = {"Tom", 18};

    memset(&infos, 0, sizeof(infos));

    register_info("int", print_int);
    register_info("string", print_string);
    register_info("person_t", print_person);

    print_text(&int_a, "int");
    print_text(&str1, "string");
    print_text(&person, "person_t");

    return;
}
```

```
[xiaohei@localhost share]$ gcc test.c && ./a.out
10
bdcom
name:Tom age:18
```

Implementation:

```c
#include<stdio.h>
#include<string.h>

typedef struct Person
{
    char name[50];
    int age;
} person_t;

// Define a structure to store function info
typedef struct {
    char type[20];
    void (*print_func)(void*);
} func_info_t;

// Global array of function infos
func_info_t infos[10];
int info_count = 0;

void print_int(void* data){
    int* value = (int*)data;
    printf("%d\n", *value);
}
```

```c
void print_string(void* data){
    char* str = (char*)data;
    printf("%s\n", str);
}

void print_person(void* data){
    person_t* person = (person_t*)data;
    printf("Name: %s, Age: %d\n", person->name, person->age);
}

void register_info(const char* type, void (*func)(void*)){
    strcpy(infos[info_count].type, type);
    infos[info_count].print_func = func;
    info_count++;
}

void print_text(void* data, const char* type){
    for(int i = 0; i < info_count; i++){
        if(strcmp(infos[i].type, type) == 0){
            infos[i].print_func(data);
            return;
        }
    }
    printf("Unknown type: %s\n", type);
}

void main(void)
{
    int int_a = 10;
    char str1[] = "bdcom";
    person_t person = {"Tom", 18};

    memset(&infos, 0, sizeof(infos));

    register_info("int", print_int);
    register_info("string", print_string);
    register_info("person_t", print_person);

    print_text(&int_a, "int");
    print_text(&str1, "string");
    print_text(&person, "person_t");

    return;
}
```

output:

```
10
bdcom
Name: Tom, Age: 18
PS C:\Users\Admin\Documents\bdcom_coding_zone\pointer_to_func>
```