Practice Questions – Structures

1. findMiddleAge
2. complexNumber
3. book
4. customer
5. employee

1. (**findMiddleAge**) Write a function that takes in an array of three persons, finds the person whose age is the middle one of the three persons, and returns the name and age of that person to the caller. For example, if the array is `{{"Tom",18},{"John",19}, {"Jim",20}}`, then the person `John` and his age will be returned. The structure Person is defined below:

```
typedef struct {
   char name[20];
   int age;
} Person;
```

The function prototype is given below:

```
Person findMiddleAge(Person *p);
```

In addition, you are also required to write another function to read the three persons' information. The input data are passed to the calling function via the pointer parameter `p`. The function prototype is given below:

```
void readData(Person *p);
```

A sample program template is given below to test the functions:

```
#include <stdio.h>
typedef struct {
   char name[20];
   int age;
} Person;
void readData(Person *p);
Person findMiddleAge(Person *p);
int main()
{
   Person man[3], middle;

   readData(man);
   middle = findMiddleAge(man);
   printf("findMiddleAge(): %s %d\n", middle.name, middle.age);
   return 0;
}
void readData(Person *p)
{
   /* Write your program code here */
}
Person findMiddleAge(Person *p)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter person 1:
john 23
Enter person 2:
```

```
peter 56
Enter person 3:
mary 31
findMiddleAge(): mary 31
```

(2) Test Case 2:

```
Enter person 1:
vincent 11
Enter person 2:
raymond 22
Enter person 3:
alex 12
findMiddleAge(): alex 12
```

2. (**complexNumber**) A structure called complex is defined to represent a complex number. Each complex number consists of the real and imaginary parts as follows:

```
typedef struct {
   double real;
   double imag;
} Complex;
```

Write C functions that perform addition, subtraction, multiplication and division operations on two complex numbers. The function prototypes are given as follows:

```
Complex add(Complex c1, Complex c2);
Complex mul(Complex c1, Complex c2);
Complex sub(Complex *c1, Complex *c2);
Complex div(Complex *c1, Complex *c2);
```

Write a C program that handles complex number arithmetic using the arithmetic functions implemented. The program will read the choice of operation (e.g. addition, subtraction, multiplication, division and quit) and two complex numbers, and display the result.

Some test input and output sessions are given below:

(1) Test Case 1:

```
Complex number operations:
1 - addition
2 - subtraction
3 - multiplication
4 - division
5 - quit
Enter your choice:
1
Enter complex number 1:
1 1
Enter complex number 2:
2 2
complex(): real 3.00 imag 3.00
Enter your choice:
5
```

(2) Test Case 2:

```
Complex number operations:
1 - addition
2 - subtraction
3 - multiplication
4 - division
5 - quit
Enter your choice:
2
Enter complex number 1:
3 3
Enter complex number 2:
```

```
2 2
complex(): real 1.00 imag 1.00
Enter your choice:
5
```

(3) Test Case 3:

```
Complex number operations:
1 - addition
2 - subtraction
3 - multiplication
4 - division
5 - quit
Enter your choice:
3
Enter complex number 1:
3 3
Enter complex number 2:
4 6
complex(): real -6.00 imag 30.00
Enter your choice:
5
```

(4) Test Case 4:

```
Complex number operations:
1 - addition
2 - subtraction
3 - multiplication
4 - division
5 - quit
Enter your choice:
4
Enter complex number 1:
1 1
Enter complex number 2:
2 2
complex(): real 0.50 imag 0.00
Enter your choice:
5
```

A sample program to test the functions is given below:

```c
#include <stdio.h>
#include <math.h>
typedef struct {
   double real;
   double imag;
} Complex;
Complex add(Complex c1, Complex c2);
Complex mul(Complex c1, Complex c2);
Complex sub(Complex *c1, Complex *c2);
Complex div(Complex *c1, Complex *c2);
int main()
{
   int choice;
   Complex input1, input2, result;

   printf("Complex number operations: \n");
   printf("1 - addition \n");
   printf("2 - subtraction \n");
   printf("3 - multiplication \n");
   printf("4 - division \n");
   printf("5 - quit \n");
   do {
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      if (choice == 5)
         return 0;
      printf("Enter Complex Number 1: \n");
```

```
            scanf("%lf %lf", &input1.real, &input1.imag);
            printf("Enter Complex Number 2: \n");
            scanf("%lf %lf", &input2.real, &input2.imag);
            switch (choice) {
               case 1: result = add(input1, input2);
                  break;
               case 2: result = sub(&input1, &input2);
                  break;
               case 3: result = mul(input1, input2);
                  break;
               case 4: result = div(&input1, &input2);
                  break;
            }
            printf("complex(): real %.2f imag %.2f\n",
                   result.real, result.imag);
      } while (choice<5);
      return 0;
}
Complex add(Complex c1, Complex c2)
{
    /* write your code here */
}
Complex sub(Complex *c1, Complex *c2)
{
    /* write your code here */
}
Complex mul(Complex c1, Complex c2)
{
    /* write your code here */
}
Complex div(Complex *c1, Complex *c2)
{
    /* write your code here */
}
```

3. (**book**) Write a program that processes an array of book records. For each book record, it stores the title of the book, author name (last name and first name), date of publication and publisher. In the program, it reads in each book's information, and then prints the book information on the display. The functions **readBook()** and **printBook()** are used for the reading and printing of each book record. The program should repeatedly read in book records from the user and print the book informtion on the screen until all book records are read. The prototypes of the two functions are given below:

```
void readBook(Booktype *book);
void printBook(Booktype book);
```

The structure definition for `Booktype` is given below:

```
typedef  struct {
   char title[81];
   char lastname[81];
   char firstname[81];
   char publisher[81];
   int day, month, year;
} Booktype;
```

A sample program template is given below to test the functions:

```
#include <stdio.h>
#define SIZE 20
typedef  struct {
   char title[81];
   char lastname[81];
   char firstname[81];
   char publisher[81];
   int day, month, year;
```

```
} Booktype;
void readBook(Booktype *book);
void printBook(Booktype book);
int main(){
   Booktype book[SIZE];
   char repeat = 'y',size=0;

   do {
      readBook(&book[size]);
      printf("The book information:\n");
      printBook(book[size]);
      printf("Continue ('y' or 'n'): ");
      scanf("%c", &repeat);
      size++;
   } while (repeat == 'y');
}
void readBook(Booktype *book)
{
    /* write your code here */
}
void printBook(Booktype book)
{
    /* write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter the title of the book:
C Programming
Enter the author's first name:
Daniel
Enter the author's last name:
Liang
Enter date as (dd-mm-yy):
12-12-12
Enter the publisher name:
Pearson
The book information:
Title: C Programming
Author: Daniel Liang
Date: 12-12-12
Publisher: Pearson
Continue ('y' or 'n'):
n
```

(2) Test Case 2:

```
Enter the title of the book:
C Programming
Enter the author's first name:
Daniel
Enter the author's last name:
Liang
Enter date as (dd-mm-yy):
12-12-12
Enter the publisher name:
Pearson
The book information:
Title: C Programming
Author: Daniel Liang
Date: 12-12-12
Publisher: Pearson
Continue ('y' or 'n'):
```

```
y
Enter the title of the book:
Java Programming
Enter the author's first name:
Daniel
Enter the author's last name:
Liang
Enter date as (dd-mm-yy):
11-11-11
Enter the publisher name:
Pearson
The book information:
Title: Java Programming
Author: Daniel  Liang
Date: 11-11-11
Publisher: Pearson
Continue ('y' or 'n'):
n
```

4. (**customer**) Write a C program that repeatedly reads in customer data from the user and prints the customer data on the screen until the customer name `"End Customer"` (i.e., first_name last_name) is read. Your program should include the following two functions: the function `nextCustomer()` reads and returns a record for a single customer to the caller via a pointer parameter ***`acct`***, and the function `printCustomer()` takes a parameter ***`acct`*** and then prints the customer information. The prototypes of the two functions are given below:

```
void nextCustomer(struct account *acct);
void printCustomer(struct account acct);
```

The structure definition for **`struct account`** is given below:

```
struct account {
   struct
   {
      char lastName[10];
      char firstName[10];
   } names;
   int accountNum;
   double balance;
};
```

A sample program template is given below to test the functions:

```
#include <stdio.h>
#include <string.h>
struct account {
   struct
   {
      char lastName[10];
      char firstName[10];
   } names;
   int accountNum;
   double balance;
};
void nextCustomer(struct account *acct);
void printCustomer(struct account acct);
int main()
{
   struct account record;
   int flag = 0;

   do {
      nextCustomer(&record);
```

```
        if ((strcmp(record.names.firstName, "End") == 0) &&
            (strcmp(record.names.lastName, "Customer") == 0))
           flag = 1;
        if (flag != 1)
           printCustomer(record);
     } while (flag != 1);
}
void nextCustomer(struct account *acct)
{
   /* Write your program code here */
}
void  printCustomer(struct account acct)
{
   /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Enter names (firstName lastName):
SC Hui
Enter account number:
123
Enter balance:
6789.89
Customer record:
SC Hui 123 6789.89
Enter names (firstName lastName):
End Customer
```

(2) Test Case 2:

```
Enter names (firstName lastName):
SC Hui
Enter account number:
123
Enter balance:
6789.89
Customer record:
SC Hui 123 6789.89
Enter names (firstName lastName):
FY Tan
Enter account number:
13
Enter balance:
69.89
Customer record:
FY Tan 13 69.89
Enter names (firstName lastName):
End Customer
```

(3) Test Case 3:

```
Enter names (firstName lastName):
End Customer
```

5. (**employee**) Write a C program that creates an array of structures to hold the employee information below:

```
typedef struct {
   char name[40];
   char telno[10];
   int id;
   double salary;
} Employee;
```

You are required to implement the following three functions:

- The function `readin()` reads a number of persons' names and their corresponding telephone numbers, passes the data to the caller via the parameter *p*, and returns the number of names that have entered. The character `'#'` is used to indicate the end of user input.
- The function `search()` allows the user to query the array using the name field. If the name is found, the program displays the message `"Employee found at index location: x"`. The function `search()` finds the employee data of an input name `target`, and then prints the name, telephone number, id and salary on the screen. If the input name cannot be found, then it will print the error message `"Name not found"` on the screen.
- If the name is not found and the array is not full, the user can then add the name as a new record into the array. Assume that the maximum size of the array is 100. The function `addEmployee()` adds a new employee record into the array. The message `"Added at position: x"` will be printed. If the database is fill, an error message `"Database is full"` will be printed on the screen. The function returns the updated size of the array after adding the new employee record.

The prototypes of the functions are given below:

```
int readin(Employee *p);
int search(Employee *p, int size, char *target);
int addEmployee(Employee *p, int size, char *target);
```

A sample program template is given below to test the functions:

```
#include <stdio.h>
#include <string.h>
#define MAX 100
typedef struct {
   char name[40];
   char telno[40];
   int id;
   double salary;
} Employee;
int readin(Employee *p);
void printEmp(Employee *p, int size) ;
int search(Employee *p, int size, char *target);
int addEmployee(Employee *p, int size, char *target);
int main()
{
   Employee emp[MAX];
   char name[40];
   int size, choice, result;

   printf("Select one of the following options: \n");
   printf("1: readin()\n");
   printf("2: search()\n");
   printf("3: addEmployee()\n");
   printf("4: print()\n");
   printf("5: exit()\n");
   do {
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      switch (choice) {
         case 1:
            size = readin(emp);
            break;
         case 2:
            printf("Enter search name: \n");
            scanf("\n");
            gets(name);
            result = search(emp,size,name);
```

```
            if (result != 1)
               printf ("Name not found!\n");
            break;
         case 3:
            printf("Enter new name: \n");
            scanf("\n");
            gets(name);
            result = search(emp,size,name);
            if (result != 1)
               size = addEmployee(emp, size, name);
            else
               printf("The new name has already existed in the database\n");
            break;
         case 4:
            printEmp(emp, size);
            break;
         default:
            break;
      }
   } while (choice < 5);
   return 0;
}
int readin(Employee *p)
{
   /* write your code here */
}
void printEmp(Employee *p, int size)
{
   /* write your code here */
}
int search(Employee *p, int size, char *target)
{
   /* write your code here */
}
int addEmployee(Employee *p, int size, char *target)
{
   /* write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Select one of the following options:
1: readin()
2: search()
3: addEmployee()
4: print()
5: exit()
Enter your choice:
1
Enter name:
Hui Siu Cheung
Enter tel:
12345678
Enter id:
11
Enter salary:
123.45
Enter name:
#
Enter your choice:
4
The current employee list:
Hui Siu Cheung 12345678 11 123.45
```

```
Enter your choice:
5
```

(2) Test Case 2:

```
Select one of the following options:
1: readin()
2: search()
3: addEmployee()
4: print()
5: exit()
Enter your choice:
1
Enter name:
Hui Siu Cheung
Enter tel:
12345678
Enter id:
11
Enter salary:
123.45
Enter name:
Kenny B
Enter tel:
23456789
Enter id:
12
Enter salary:
1234.45
Enter name:
#
Enter your choice:
4
The current employee list:
Hui Siu Cheung 12345678 11 123.45
Kenny B 23456789 12 1234.45
Enter your choice:
2
Enter search name:
Kenny B
Employee found at index location: 1
Kenny B 23456789 12 1234.45
Enter your choice:
5
```

(3) Test Case 3:

```
Select one of the following options:
1: readin()
2: search()
3: addEmployee()
4: print()
5: exit()
Enter your choice:
1
Enter name:
Hui Siu Cheung
Enter tel:
12345678
Enter id:
11
Enter salary:
123.45
Enter name:
Kenny B
Enter tel:
```

```
23456789
Enter id:
12
Enter salary:
1234.45
Enter name:
#
Enter your choice:
4
The current employee list:
Hui Siu Cheung 12345678 11 123.45
Kenny B 23456789 12 1234.45
Enter your choice:
2
Enter search name:
Kenny BB
Name not found!
Enter your choice:
5
```

(4) Test Case 4:

```
Select one of the following options:
1: readin()
2: search()
3: addEmployee()
4: print()
5: exit()
Enter your choice:
1
Enter name:
Hui Siu Cheung
Enter tel:
12345678
Enter id:
11
Enter salary:
123.45
Enter name:
Kenny B
Enter tel:
23456789
Enter id:
12
Enter salary:
1234.45
Enter name:
#
Enter your choice:
4
The current employee list:
Hui Siu Cheung 12345678 11 123.45
Kenny B 23456789 12 1234.45
Enter your choice:
3
Enter new name:
Kenny Tan
Enter tel:
12344321
Enter id:
13
Enter salary:
2345.67
Added at position: 2
```

```
Enter your choice:
4
The current employee list:
Hui Siu Cheung 12345678 11 123.45
Kenny B 23456789 12 1234.45
Kenny Tan 12344321 13 2345.67
Enter your choice:
5
```