QUESTION 1

(a) Create a structure called person which is made up of a string for the name and an integer for the age.
(b) Create an array variable for the structure that will hold for 5 persons.
(c) Read the name and age values from the keyboard for these 5 persons in main().
(d) Write a function called displayOldest to print the oldest person's name. (Function will check each person's age and print the person's name whose age is the maximum of all).

```
void displayOldest(struct person Persons[ ], int size); // function prototype
```

sample solution:

```
#include <stdio.h>

// (a)
struct person{
    char name[40];
    int age;
};

void displayOldest(struct person Persons[ ], int size);
int main()
{
    int i;

    // (b)
    struct person myArray[5];

    // (c)
    printf("Enter names and ages for 5 persons:\n");
    for(i=0;i<5;i++){
        scanf("%s",myArray[i].name);
        scanf("%d",&myArray[i].age);
    }

    displayOldest(myArray,5);
    return 0;
}

// (d)
void displayOldest(struct person Persons[ ], int size){
    int i;
    struct person oldest=Persons[0];
    for(i=0;i<size;i++){
        if(oldest.age<Persons[i].age){
            oldest=Persons[i];
        }
    }
    printf("Oldest person's name: %s \n",oldest.name);
    return;
}
```

QUESTION 2

(a) Create a structure called ***Student*** which is made up of an integer for the ***ID*** and an integer for the ***age***.
(b) Write a function called ***displayEldest*** to print the oldest student's *ID* in a given array of students. (Function will check each student's *age* and print the *ID* of the student whose *age* is the maximum of all).

void displayEldest(struct Student myStudents[ ], int size); // function prototype

(c) Write a ***main( )*** function and create an array variable for the structure that will hold for 20 students.
(d) Read the *ID* and *age* values from the keyboard for these 20 students in ***main( )*** and print the *ID* of the student whose *age* is maximum.

sample solution:

```
#include <stdio.h>

// (a)
struct Student{
   int ID;
   int age;
};

// (b)
void displayEldest(struct Student myStudents[ ], int size){
   int i;
   struct Student oldest=myStudents[0];
   for(i=0;i<size;i++){
      if(oldest.age<myStudents[i].age){
         oldest=myStudents[i];
      }
   }
   printf("Oldest student's ID: %d \n",oldest.ID);
   return;
}

// (c)
int main()
{
   struct Student myArray[20];

// (d)
   int i;
   printf("Enter IDs and ages for 20 students:\n");
   for(i=0;i<20;i++){
      scanf("%d",&myArray[i].ID);
      scanf("%d",&myArray[i].age);
   }
   displayEldest(myArray,20);

   return 0;
}
```

QUESTION 3

Given the following structure and variable definitions,

```
struct personal{
   char phoneNumber[11];
   char address[50];
   char city[15];
   char state[3];
   char zipCode[6];
};

struct customer{
   char lastName[15];
   char firstName[15];
   int customerNumber;
   struct personal personalRecord;
};

struct customer customerRecord;
struct customer *customerPtr;
customerPtr = &customerRecord;
```

write an expression that can be used to access the structure members in each of the following parts:

a) Member **lastName** of structure **customerRecord**.
customerRecord.lastName

b) Member **lastName** of the structure pointed to by **customerPtr**.
customerPtr→lastName

c) Member **firstName** of structure **customerRecord**.
customerRecord.firstName

d) Member **firstName** of the structure pointed to by **customerPtr**.
customerPtr→firstName

e) Member **customerNumber** of structure **customerRecord**.
customerRecord.customerNumber

f) Member **customerNumber** of the structure pointed to by **customerPtr**.
customerPtr→customerNumber

g) Member **city** of member **personalRecord** of the structure pointed to by **customerPtr**.
customerPtr→personalRecord.city

h) Member **state** of member **personalRecord** of structure **customerRecord**.
customerRecord.personalRecord.state

QUESTION 4

a. Declare a student structure containing the following three data members: stdID, name, and age.
b. Define fillStudentArray function that takes an array of students and the size of the array, and fills the information of the students. The prototype shall be as follows:

```
void fillStudentArray(struct student *array, int size);
```

c. Define minStudent function that takes an array of students and the size of the array, and returns the student who has the minimum age value.

```
struct student minStudent(struct student *array, int size);
```

d. Implement a proper main function to test the functions above.

sample solution:

```
#include <stdio.h>

#define ARRAY_SIZE 5

struct student {
    int StdID;
    char name [20];
    int age;
};

void fillStudentArray(struct student *array, int size);
struct student minStudent(struct student *array, int size);

int main(){
    struct student array[ARRAY_SIZE];
    struct student min;

    fillStudentArray(array, ARRAY_SIZE);
    min = minStudent(array, ARRAY_SIZE);

    printf("Youngest Student :\n");
    printf("ID :\t%d\nName :\t%s\nAge :\t%d\n",min.StdID, min.name, min.age);

    return 0;
}

void fillStudentArray(struct student *array, int size){
    int i;
    for (i = 0; i < size; i++){
        printf("%d. student (ID name age): ",i+1);
        scanf("%d %s %d", &array[i].StdID, array[i].name, &array[i].age);
    }

    return;
}
```

```
struct student minStudent(struct student *array, int size){
    struct student youngestStd;
    int i;
    youngestStd = array[0];

    for(i = 1; i < size; i++){

        if (array[i].age < youngestStd.age)
            youngestStd = array[i];
    }

    return youngestStd;
}
```

QUESTION 5

Write a program that uses an *updateTime* function to find the correct time at 30 seconds after the *time* given as the argument. The function shall use *call-by-reference* concept to modify the given *time* argument.

```
struct Time {

        int hour, minutes, seconds;

};

void updateTime( struct Time * ); /* function prototype */
```

For example:

If current time is 14:35:45, then updated time will be 14:36:15.

If current time is 16:59:52, updated time will be 17:00:22.

sample solution:

```
#include <stdio.h>

struct Time {
    int hour, minutes, seconds;
};

void updateTime( struct Time * ); /* function prototype */

int main()
{
    struct Time currTime;
    printf("Current Time: hour minute second (enter such as follows: 12 53 36):\n");
    scanf("%d %d %d",&currTime.hour,&currTime.minutes,&currTime.seconds);

    printf("\ncurrent time is %d:%d:%d, ",currTime.hour,currTime.minutes,currTime.seconds);

    updateTime(&currTime);

    printf("updated time is %d:%d:%d \n",currTime.hour,currTime.minutes,currTime.seconds);
```

```
    return 0;
}

void updateTime( struct Time *currT){

    int minute, second;

    second = currT->seconds + 30;
    currT->seconds = second % 60;
    minute = currT->minutes + (second / 60);
    currT->minutes = minute % 60;
    currT->hour = (currT->hour + (minute / 60)) % 24;

    return;
}
```