WEEK-END ASSIGNMENT-09
Structure and Union
Operating Systems Workshop (CSE 3541)

Problem Statement:

Experiment with user-defined data types to store heterogeneous data and their processing.

Assignment Objectives:

To learn about structure & union data types and their implications in programming.

Instruction to Students (If any):

Students are required to write his/her own program by avoiding any kind of copy from any sources. Additionally, They must be able to realise the outcome of that question in relevant to systems programming. You may use additional pages on requirement.

Programming/ Output Based Questions:

1. Select the invalid member of the following structure;

```
struct oswcourse{
   int secid;
   float avgm;
   char present;
   int *marks();
   int teacher();
}o1,o2;
```

Output with explanation

The invalid member of the structure is int *marks().

In C, function pointers are not allowed directly within structures.
The structure can only contain data members, which are variables of
fundamental types or pointers to those types.
Function pointers are not considered data members.

2. Detect any invalid member present in the given structure;

```
struct date{
    int m,d,y;
};
struct stud{
   char name[20];
   struct stud *p;
   struct date *d;
   int (*)fun(int, int);
};
```

Output with explanation

Members	Reasons
d	Function pointers are not allowed directly within structures.
fun	Function pointers are not allowed directly within structures.
name	Arrays cannot be members of structures.
p	Function pointers are not allowed directly within structures.

3. The following structure template is allowed or not in ANSI C.

```
struct person{
    int a;
    struct health{
        int a;
    }h;
};
```

Output with explanation

Thestructure template is allowed in ANSI C.

The code defines a structure named person with two members:

a: An integer variable
h: A structure named health with one member:
a: An integer variable
This type of nested structure is allowed in ANSI C.

4. The following declaration is correct or wrong.

```
struct person{
    int a;
    union health{
        int w;
    }h;
};
```

Output with explanation

The declaration is correct.

Unions can have multiple members, but only one member can be active at a time.

5. The following declaration is correct or wrong.

```
union person{
    int a;
    struct health{
        int e;
    }h;
};
```

Output with explanation

The declaration is wrong because it combines a union and a structure in a way that is not allowed in C.

6. Check the declaration of the structure. Write a valid conclusion whether `Line-5` can be valid member or not.

```
struct person{
    int ht;
    float wt;
    char color;
    struct person p; /*Line- 5 */
};
```

Output with explanation

Yes, Line-5 can be a valid member of the structure. This is because C allows structures to contain members of their own type, creating a self-referential structure. This enables you to represent hierarchical relationships or linked lists of data.

7. Write valid or invalid form of the followings.

```
(1) union{....}u;
(2) union u{......};
(3) struct{.....}s;
(4) struct s{.....};
```

Output with explanation

all are valid form .

8. Decide the output of the code snippet;

```
int main(){
   struct student{
      int h;
      int w;
      int m;
   };
   struct student s1={20,40,50};
   struct student *ptr=&s1;
   printf("%d\n",*((int *)ptr+2));
   return 0;
}
```

Output and reason▼

50

9. Find the output of the code snippet;

```
struct s{int *p;};
int main(){int a=200;struct s s1;
  s1.p=&a;   *(s1.p)=*(s1.p)+100;
  printf("%d %d\n",a,*(s1.p));
  return 0;}
```

Output and reason▼

300 300

10. Draw the node connectivity of the structure **s1** and determine the output of the code snippet that simulates the array of structures and also the self-referential structure;

```
int main(){
  struct s1{
   char *z;
   int i;
   struct s1 *p;
};
struct s1 a[]={{"SOA",1,a+1},
               {"ITER",4,a+2},
               {"CSE",5,a}
              };
struct s1 *ptr=a;
printf("%s%s%s\n",a[0].z,a[1].z,a[2].z);
printf("%s%s%s",(*ptr).z, ptr→z,a[2].p→z);
return 0;
}
```

Draw figure and Output▼

```
SOAITERCSE
SOASOASOA
```

11. Draw the node connectivity of the structure **s1** and determine the output of the code snippet that simulates the array of structures and also the self-referential structure;

```
intmain(){
  struct s1{
   char *z;
   int i;
   struct s1 *p;
};
 struct s1 a[]={{"SOA",1,a+1},
                  {"ITER",2,a+2},
                  {"CSE",3,a}};
 struct s1 *ptr=a;
 printf("%s\n", ++(ptr→z));
 printf("%s\n", a[(++ptr)→i].z);
 printf("%s\n",a[--(ptr→p→i)].z);
 printf("%d\n",--a[2].i);
 return 0;
}
```

Draw figure and Output▼

```
OA
CSE
CSE
1
```

12. An initialization of array of structures given in the following code snippet. Find the output with pointer manipulation and operator precedence rules.

```
int main(){
  struct test{
   int i;
   char *c;
};
struct test st[]={5, "Cse-Engg",
                  4, "computer",
                  6, "Electrical",
                  8, "Mechnical",
                  7, "All-Engg"
                };
struct test *p=st;
printf("%s\n", ++(p++ →c));
printf("%c\n",*p++ →c);
printf("%d\n",++p→i);
printf("%s\n",p[0].c);
printf("%s\n",p→c);
return 0;}
```

Output▼

```
computer
c
8
Mechanical
Mechanical
```

13. Conclude the output of the code snippet based on pointer and operator precedence on a nested structure case.

```
int main(){
 struct out{
      char ch[10];
      char *str;
 };
 struct b{
       char *c;
       struct out o;
 };
 struct b s2={"ODISHA", "KHURDA","JOYDEV"};
 printf("%s %s %s\n",s2.c,s2.o.str,s2.o.ch);
 printf("%s %s\n",++s2.c,++s2.o.str);
 return 0;
}
```

Output and reason▼

ODISHA JOYDEV KHURDA
DISHA OYDEV

14. Find the output of the code snippet;

```
int main(){
   union unit{
   int marks;
   int roll;
}s1,s2;
 s2.roll=23;
 s1.marks=60;
 printf("%d..%d\n",s1.marks,s2.roll);
 return 0;
}
```

Output and reason▼

60..23

15. Find the output of the code snippet;

```
int main(){
   union unit{
   int marks;
   int roll;
}s1,s2;
 s2.roll=23;
 s2.marks=60;
 printf("Check memory alloc  for union\n");
 printf("%d..%d\n",s2.marks,s2.roll);
 return 0;
}
```

Output and reason▼

Check memory alloc for union
60..60

16. Declare two variable of the structure type `planet_t`

```
typedef struct{
   char name[30];
   double diameter;
   int moons;
   double or_time,ro_time;
}planet_t;
```

Declaration▼

planet_t planet1;
planet_t planet2;
or
planet_t planets[2];

17. Initialize one of the variable of the question-16 structure with values ”jupiter”, 142.34, 16, 11.9, 9.23;

Initialization▼

```
planet_t planet1;
planet1.name = "Jupiter";
planet1.diameter = 142.34;
planet1.moons = 16;
planet1.or_time = 11.9;
planet1.ro_time = 9.23;
```

18. Declare a pointer to the structure type `planet_t` and initialize the structure components with the help of the pointer.

Initialization▼

```
planet_t *planet;

planet = malloc(sizeof(planet_t));

planet->name = "Jupiter";
planet->diameter = 142.34;
planet->moons = 16;
planet->or_time = 11.9;
planet->ro_time = 9.23;
```

19. Assuming a 24-hr clock and the structure type `time_t` is defined below.

```
typedef struct{ int hour, minute, second; };
```

Write a program that contains a function `new_time(...)` that returns as its value an updated time based the original time of day and the number of seconds that have elapsed since the previous update time. if time now were 21-58-32 and elapsed second 97, the original time now is 22-00-09. The function prototype to compute the new time is **time_t new_time(time_t time_of_day, int elapsed_secs);**

Code here▼

```
#include <stdio.h>


typedef struct {
  int hour;
  int minute;
  int second;
} time_t;


time_t new_time(time_t time_of_day, int elapsed_secs) {
  time_t updated_time;

  updated_time.second = time_of_day.second + elapsed_secs;

  if (updated_time.second >= 60) {
    updated_time.minute += updated_time.second / 60;
    updated_time.second %= 60;
  }
```

Code here▼

```
 if (updated_time.minute >= 60) {
    updated_time.hour += updated_time.minute / 60;
    updated_time.minute %= 60;
  }

  updated_time.hour %= 24;

  return updated_time;
}

int main() {
  time_t current_time = {21, 58, 32};
  int elapsed_seconds = 97;

  time_t updated_time = new_time(current_time, elapsed_seconds);

  printf("Original time: %02d:%02d:%02d\n", current_time.hour, current_time.minute,
  current_time.second);
  printf("Elapsed seconds: %d\n", elapsed_seconds);
  printf("Updated time: %02d:%02d:%02d\n", updated_time.hour, updated_time.minute,
  updated_time.second);

  return 0;
}
```

20. Define a structure type **`auto_t`** to represent an automobile. Include components for the make and model (strings), the odometer reading, the manufacture and purchase dates (use another user-defined type called **`date_t`**), and the gas tank (use a user-defined type **`tank_t`** with components for tank capacity and current fuel level, giving both in gallons). Write I/O functions **`scan_date`** , **`scan_tank`** , **`scan_auto`** , **`print_date`** , **`print_tank`** , and **`print_auto`** , and also write a driver function that repeatedly fills and displays an auto structure variable until **`EOF`** is encountered in the input file. Here is a small data set to try:

```
Mercury Sable 99842 1 18 2001 5 30 1991 16 12.5
Mazda Navajo 123961 2 20 1993 6 15 1993 19.3 16.7
```

Code here▼

```
#include <stdio.h>

// Date structure
typedef struct {
  int month;
  int day;
  int year;
} date_t;

// Tank structure
```

Code here▼

```
typedef struct {
  float capacity;
  float level;
} tank_t;

typedef struct {
  char make[20];
  char model[20];
  int odometer;
  date_t manufactured_date;
  date_t purchased_date;
  tank_t gas_tank;
} auto_t;

void scan_date(date_t *date) {
  printf("Enter month: ");
  scanf("%d", &date->month);
  printf("Enter day: ");
  scanf("%d", &date->day);
  printf("Enter year: ");
  scanf("%d", &date->year);
}

void scan_tank(tank_t *tank) {
  printf("Enter tank capacity: ");
  scanf("%f", &tank->capacity);
  printf("Enter current fuel level: ");
  scanf("%f", &tank->level);
}

void scan_auto(auto_t *auto) {
  printf("Enter make: ");
  scanf("%s", auto->make);
  printf("Enter model: ");
  scanf("%s", auto->model);
  printf("Enter odometer reading: ");
  scanf("%d", &auto->odometer);
  printf("Enter manufactured date:\n");
  scan_date(&auto->manufactured_date);
  printf("Enter purchased date:\n");
  scan_date(&auto->purchased_date);
  printf("Enter tank information:\n");
  scan_tank(&auto->gas_tank);
}


void print_date(date_t *date) {
  printf("%d/%d/%d", date->month, date->day, date->year);
}

void print_tank(tank_t *tank) {
  printf("Tank capacity: %.2f gallons\n", tank->capacity);
  printf("Current fuel level: %.2f gallons\n", tank->level);
}

void print_auto(auto_t *auto) {
  printf("Make: %s\n", auto->make);
  printf("Model: %s\n", auto->model);
  printf("Odometer reading: %d\n", auto->odometer);
  printf("Manufactured date: ");
  print_date(&auto->manufactured_date);
  printf("\n");
  printf("Purchased date: ");
  print_date(&auto->purchased_date);
  printf("\n");
```

Code here▼

```
printf("Tank information:\n");
  print_tank(&auto->gas_tank);
  printf("\n");
}

int main() {
  auto_t my_auto;

  while (1) {
    scan_auto(&my_auto);


    printf("Enter amount of fuel to add: ");
    float fuel_added;
    scanf("%f", &fuel_added);
    my_auto.gas_tank.level += fuel_added;
    if (my_auto.gas_tank.level > my_auto.gas_tank.capacity) {
      printf("Warning: Tank overflow!\n");
      my_auto.gas_tank.level = my_auto.gas_tank.capacity;
    }

    print_auto(&my_auto);


    if (feof(stdin)) {
      break;
    }
  }

  return 0;
}
```

21. Numeric addresses for computers on the international network Internet are composed of four parts, separated by periods, of the form

```
xx.yy.zz.mm
```

where **`xx, yy, zz`** , and mm are positive integers. Locally, computers are usually known by a nickname as well. You are designing a program to process a list of Internet addresses, identifying all pairs of computers from the same locality. Create a structure type called **`address_t`** with components for the four integers of an Internet address and a fifth component in which to store an associated nickname of ten characters. Your program should read a list of up to 100 addresses and nicknames terminated by a sentinel address of all zeros and a sentinel nickname.

```
Sample Data
111.22.3.44 platte
555.66.7.88 wabash
111.22.5.66 green
0.0.0.0 none
```

The program should display a list of messages identifying each pair of computers from the same locality, that is, each pair of computers with matching values in the first two components of the address. In the messages, the computers should be identified by their nicknames.

```
Example Message
Machines platte and green are on the same local
   network.
```

Follow the messages by a display of the full list of addresses and nicknames. Include in your program a **`scan_address`** function, a **`print_address`** function, and a **`local_address`** function. Function **`local_address`** should take two address structures as input parameters and return 1 (for true) if the addresses are on the same local network, and 0 (for false) otherwise.

Code here▼

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


typedef struct {
   int xx, yy, zz, mm;
   char nickname[11];
} address_t;


void scan_address(address_t *address) {
   scanf("%d.%d.%d.%d %s", &address->xx, &address->yy, &address->zz, &address->mm, address
->nickname);
}


void print_address(address_t address) {
   printf("%d.%d.%d.%d %s\n", address.xx, address.yy, address.zz, address.mm, address.nickname);
}


int local_address(address_t address1, address_t address2) {
   return (address1.xx == address2.xx && address1.yy == address2.yy);
}

int main() {
   address_t addresses[100];
   int count = 0;
```

Code here▼

```
printf("Enter addresses and nicknames (up to 100) terminated by 0.0.0.0 none:\n");
    do {
        scan_address(&addresses[count]);
        count++;
    } while (count < 100 && !(addresses[count - 1].xx == 0 && addresses[count - 1].yy == 0));


    printf("\nPairs of computers on the same local network:\n");
    for (int i = 0; i < count - 1; i++) {
        for (int j = i + 1; j < count; j++) {
            if (local_address(addresses[i], addresses[j])) {
                printf("Machines %s and %s are on the same local network.\n", addresses[i].nickname,
         addresses[j].nickname);
            }
        }
    }


    printf("\nList of addresses and nicknames:\n");
    for (int i = 0; i < count - 1; i++) {
        print_address(addresses[i]);
    }

    return 0;
}
```