Department of Computer Science and Engineering
Institute of Technical Education & Research, SOA, Deemed to be University

MINOR ASSIGNMENT-07

Practical Programming with C (CSE 3544)

Publish on: 07-12-2024 **Submission on:** 10-12-2024
Course Outcome: CO_4 **Program Outcome:** PO_5 **Learning Level:** L_5

Problem Statement:

Experiment with C structures and unions to handle heterogeneous data and their processings.

Assignment Objectives:

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

Answer the followings:

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

Int *marks() and int teacher() are not allowed. function declarations are not directly allowed inside C structure.

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

int(*)fun(int,int) is not allowed because it's not a right way to write a pointer to function.

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

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

Output with explanation

Yes, it's allowed in ANSI C. This is nested structure where person is an outer structure and heath is an inner structure.

4. The following declaration is correct or wrong.

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

Output with explanation

Yes, the above declaration is correct. Because union is allowed to be a member of a structure.

5. The following declaration is correct or wrong.

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

Output with explanation

Yes, Because structure is allowed to be a member of an union.

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

No, line 5 can't be a valid member because this is a recursive structure which will continuously create new instances of p.

7. Write valid or invalid form of the followings.

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

Output with explanation

(1) Valid (anonymous union)
(2) Valid (named union)
(3) Valid (anonymous structure)
(4) Valid (named structure)

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

Ptr is holding the address of structre s1. Adding 2 to it takes us to the third member (m). Dereferencing the variable gives us the value 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

Giving the address of a to structure s1 as member, then manipulating the value in that address will change value of both a and *(s1.p).

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;

Department of Computer Science and Engineering


```
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▼

O/P
OA
CSE
CSE
2

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;

```
int main(){
  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
2

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▼

O/P
se-Engg
c
7
Electrical
Electrical

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

ture 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
Inside structure b there is an instance of structure out. Accessing structure members using . operator.

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
Simple assignment of values to both the instances of unit union members.

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▼

60..60
union allocates only one memory space for all the members. Hence, the last updated value is in that address. No matter through which member we try to access the value it'll return the same value from that address.

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▼

```
int main(){
  planet_t planet_1, plan-
  -et_2;
  return 0;}
```

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

Initialization▼

```
strcpy(planet_1.name, "Jupiter");
planet_1.diameter = 142.34;
planet_1.moons = 16;
planet_1.or_time = 11.9;
planet_1.ro_time = 9.23;
  return 0; }
```

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

Initialization▼

```
planet_t *ptr = &planet_2;
strcpy(ptr->name, "Earth");
ptr -> diameter = 200;
ptr-> moons = 1;
ptr -> or_time = 365.25;
ptr -> ro_time = 24;
   return 0;
```

19. 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>
#define MAX_ADDRESSES 100
#define MAX_NICKNAME_LENGTH 10

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

int scan_address(address_t addresses[]) {
    int count = 0;
    while (1) {
        printf("Enter address (xx.yy.zz.mm nickname): ");
        scanf("%d.%d.%d.%d%s", &addresses[count].xx, &address
            [count].yy, &address[count].zz, &addresses
            [count].mm, address[count].nickname);

        if(address[count].xx == 0 && addresses[count].yy == 0 &&
          address[count].zz == 0 && address[count].mm == 0
          && strcmp(address[count].nickname, "none") ==
          0) {
            break;
        }
        count++;
        if (count >= MAX_ADDRESSES) {
            break;
        }
    }
    return count;
}
void print_address(address_t addresses[], int count) {
    printf("\nFull list of address & nicknames:\n");
    for (int i = 0; i < count; i++) {
        printf("%d.%d.%d.%d %s\n", addresses[i].xx,
            addresses[i].yy, addresses[i].zz, address[i].mm
            addresses[i].nickname);
    }
```

Code here ▼

```
int local_address (address_t addr1, address_t addr2){
    return (addr1.xx == addr2.xx && addr1.yy ==
                                    addr2.yy);
}
int main() {
    address_t addresses [MAX_ADDRESSES];
    int count = scan_address (addresses);
    for (int i=0; i<count ; i++){
        for (int j=0; j < count ; j++) {
            i+1
            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);
            }
        }
    }
    print_address (addresses , count);
    return 0;
}
```

20. You know that a single linked list consists of several nodes that are connected through pointers. Design a program to create a singly linked list comprising integer elements for the given **n** nodes. A node contains, an integer number and a self-referential structure of the structure type **node**. Additionally, sort this linked list in ascending order.

Code here▼

```
#include <stdio.h>
#include <stdlib.h>
Struct Node{
    int data;
    Struct Node* next;
};
Struct Node* creatNode (int value){
    Struct Node* newNode = (struct Node*)malloc (sizeof
    (struct Node));
    newNode -> data = value;
    newNode -> next = NULL;
    return newNode;
}
void append (struct Node** head, int value){
    Struct Node* temp = *head;
    if (!temp){
        *head = creatNode (value);
        return;
    }
    while (temp -> next) temp = temp -> next;
    temp -> next = createNode (value);
}
void printList (struct Node* head){
    while (head){
        printf ("%d", head -> data);
        head = head -> next;
    }
    printf ("\n");
}
void sortList (struct Node* head){
    for (struct Node* i = head; i && i -> Next; i
    = i -> next){
        for (struct Node* j = head; j -> next; j = j -> next){
            if (j -> data > j -> next -> data){
```

Code here▼

```
        int temp = j->data;
        j->data = j->next->data;
        j->next->data = temp;
            }
        }
    }
int main() {
    struct Node* head = NULL;
    int n, value;
    printf("Enter the number of nodes: ");
    scanf("%d", &n);
    for(int i = 0; i < n; i++){
      printf("Enter the value for node %d: ", i+1);
      scanf("%d", &value);
      append(&head, value);
      }
      printf("Linked list before sorting: ");
      printf(head);
      sortList(head);
      printf("Linked list after sorting: ");
      printList(head);
      return 0;
      }
```