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

MINOR ASSIGNMENT-002

UNIX Systems Programming (CSE 3041)

Assignment Objectives:

To learn about pointers, indirect addressing and how to return function results through a function's arguments. To understand the differences between call-by value & call-by-reference. To understand the distinction between input, input-output and output parameters and the usages of each kind. To learn how to declare and use arrays for storing collections of values of the same type and learn about array processing in different applications.

Instruction to Students (If any):

This assignment is designed to give practice with pointers and arrays in C. Students are required to create their own programs to solve each and every question/problem as per the specification of the given question/problem to meet the basic requirement of Unix systems programming. **Students are required to ~~write the output/~~ paste the output screen shots onto their laboratory record after each question.**

Programming/ Output Based Questions:

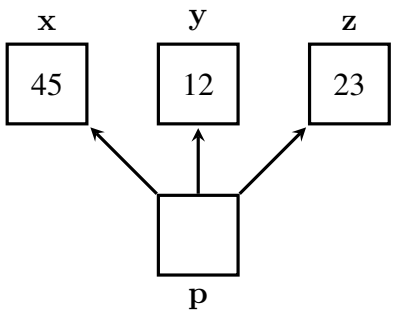
P = Write program.
S = Don't write a program, just solve it

P 1. For the given structure below, declare the variable type, and print their values;

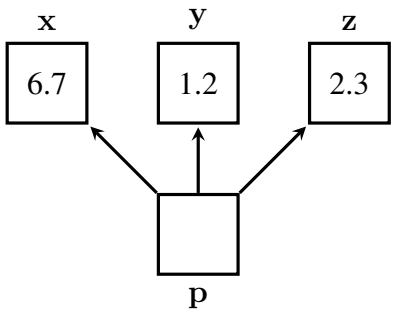
P 2. For the given structure below, declare the variable type, print their values and addresses;

P 3. Declare two integer variable and assign values to them, and print their addresses. Additionally, Swap the contents of the variables and print their addresses after swap. State whether the addresses before and after are equal or not.

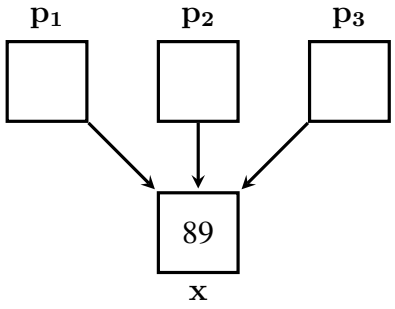
P 4. Write the C statement to declare and initialize the pointer variable for the given structure and display the values of **x**, **y** and **z** with the help of **p**.



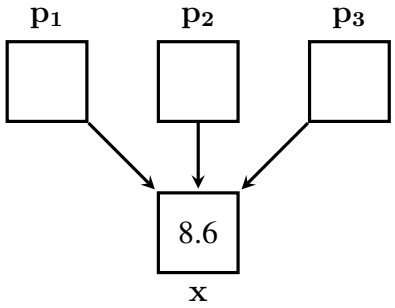
P 5. Write the C statement to declare and initialize the pointer variable for the given structure.



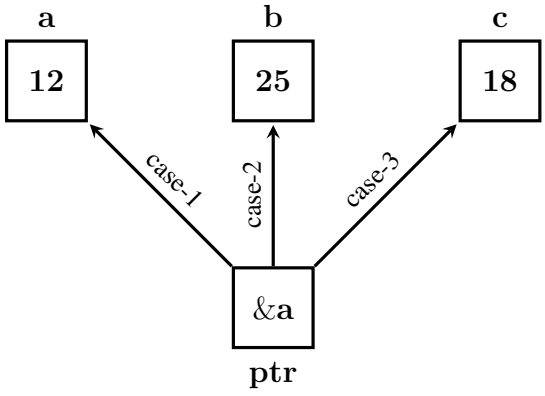
P 6. Write the C statement to declare and initialize the pointer variable for the given structure.



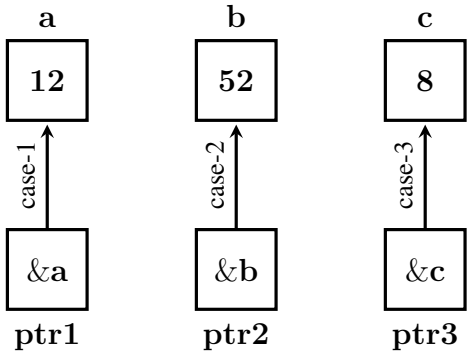
P 7. Write the C statement to declare and initialize the pointer variable for the given structure.



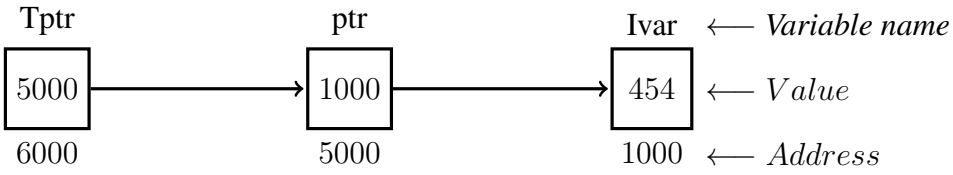
P 8. Write the C statement to declare and initialize the pointer variable for the given structure and update the values of **a**, **b** and **c** to be incremented by 10.



P 9. Write the C statement to declare and initialize the pointer variable for the given structure and update the values of **a**, **b** and **c** to be incremented by 10.

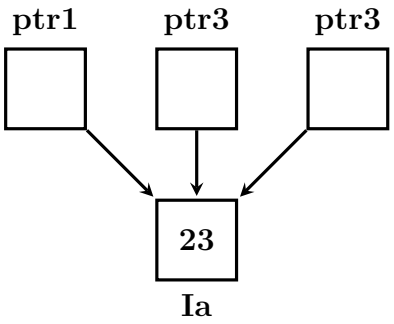


P 10. Declare and initialize the pointer variables.



P 11. Two pointers are pointing to different variable. Write the C++ statement to find the greater between **a**, and **b** through pointer manipulation.

P 12. Write the C++ statement to manipulate the value of the variable **Ia** through the pointers **ptr1**, **ptr2**, and **ptr3**.



We can use different pointers to point the same data variable. For example;

1. int **Ia**;
2. int ∗**ptr1** = &**Ia**;
3. ··· // manipulate the variable **Ia**
4. int ∗**ptr2** = &**Ia**;
5. ··· // manipulate the variable **Ia**
6. ···

S 13. Trace the execution of the following fragment at **`line -1`**.

```
int m = 10, n = 5;
int *mp, *np;
mp = &m;
```

```
np = &n;
*mp = *mp + *np;
*np = *mp - *np;
printf("%d %d\n%d %d\n", m, *mp, n, *np); /*line-1 */
```

14. Given the declarations

```
int m = 25, n = 77;
char c = '*';
int *itemp;
describe the errors in each of the following statements.
m = &n;
itemp = m;
*itemp = c;
*itemp = &c;
```

15. Write a prototype for a function **sum_n_avg** that has three type double input parameters and two output parameters. The function computes the sum and the average of its three input arguments and relays its results through two output parameters.

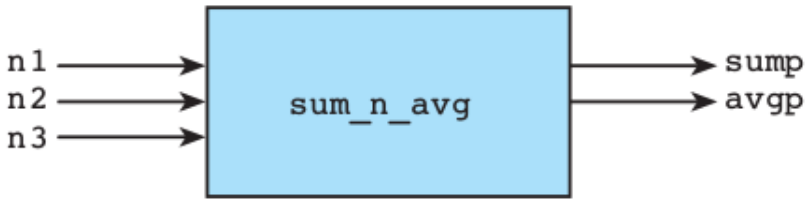
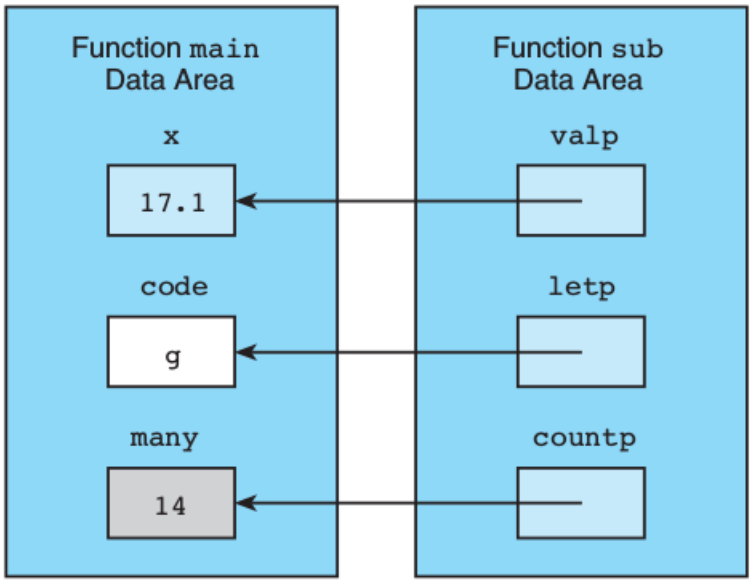


Figure 1: Question-15

16. Given the memory setup shown, fill in the chart by indicating the data type and value of each reference as well as the name of the function in which the reference would be legal. Describe pointer values by



referring to cell attributes. For example, the value of **valp** would be "pointer to color-shaded cell",

and the value of **&many** would be "pointer to gray-shaded cell".

Reference	Where Legal	Data Type	Value
valp	sub	double *	pointer to color-shaded cell
&many			
code			
&code			
countp			
*countp			
*valp			
letp			
&x			

P17. The following code fragment is from a function preparing to call **sum_n_avg** (see question-15). Complete the function call statement.

```
{
double one, two, three, sum_of_3, avg_of_3;
printf("Enter three numbers> ");
scanf("%lf%lf%lf", &one, &two, &three);
sum_n_avg(_________________________);
. . .
}
```

Define the function **sum_n_avg** whose prototype you wrote in question-15. The function should compute both the sum and the average of its three input parameters and relay these results through its output parameters. Design the complete C code to get the desire result.

P18. Write a program to use the idea of **multiple calls to a function with input/output parameters** to sort 6 integer numbers in ascending order without using any sorting algorithms. The prototype of the function to be used in your program to sort the numbers is given as **void arrange(int *, int *);** and also draw the data areas of calling function and **arragne()** function for the first function call **arrange(....)**.

Sample Run

```
printf("Enter SIX numbers separated by blanks> ");
12 3 56 8 20 654
/* Displays results */
printf("The numbers in ascending order are: %d %d %d %d %d
    %d\n",n1, n2, n3,n4,n5,n6);
3 8 12 20 56 654
```

S19. Show the table of values for x , y , and z that is the output displayed by the following program.

```
#include <stdio.h>
void sum(int a, int b, int *cp);
int main(void){
int x, y, z;
x = 7; y = 2;
printf("x y z\n\n");
sum(x, y, &z);
printf("%4d%4d%4d\n", x, y, z);
sum(y, x, &z);
printf("%4d%4d%4d\n", x, y, z);
sum(z, y, &x);
printf("%4d%4d%4d\n", x, y, z);
sum(z, z, &x);
printf("%4d%4d%4d\n", x, y, z);
sum(y, y, &y);
printf("%4d%4d%4d\n", x, y, z);
return (0);
}
```

```
void sum(int a, int b, int *cp)
{
*cp = a + b;
}
```

20. (a) Classify each formal parameter of **double_trouble** and **trouble** as input, output, or input/output.

(b) What values of x and y are displayed by this program? (Hint: Sketch the data areas of **main** , **trouble** , and **double_trouble** as the program executes.)

```
void double_trouble(int *p, int y);
void trouble(int *x, int *y);
int main(void){
  int x, y;
  trouble(&x, &y);
  printf("x = %d, y = %d\n", x, y);
  return (0);
}
void double_trouble(int *p, int y){
  int x;
  x = 10;
  *p = 2 * x - y;
}
void trouble(int *x, int *y){
  double_trouble(x, 7);
  double_trouble(y, *x);
}
```

P 21. Since communications channels are often noisy, numerous ways have been devised to ensure reliable data transmission. One successful method uses a checksum. A checksum for a message can be computed by summing the integer codes of the characters in the message and finding the remainder of this sum divided by 64. The integer code for a space character is added to this result to obtain the checksum. Since this value is within the range of the dis- playable characters, it is displayed as a character as well. Write a program that accepts single-line messages ending with a period and displays the checksum character for each message. Your program should continue displaying check-sums until the user enters a line with only a period.

P 22. Write a program in C to add numbers using function via call by reference.

Sample run

```
Input the first number : 5
Input the second number : 6
The sum of 5 and 6  is 11
```

P 23. Write a program in C to find the maximum and minimum number in an array having 5 elements.

Sample run

```
Input 5 elements in the array:
arr[0]: 45
arr[1]: 34
arr[2]: 12
arr[3]: 56
arr[4]: 23
Maximum element is: 56
Minimum element is: 12
```

P 24. Write a program in C for a menu driven calculator having 4 functionalities Addition, Subtraction, Multiplication and Division. All the functionalities should be implemented in functions which should be called via function pointers.

P 25. Write a program in C to read a sentence and replace lowercase characters by uppercase and vice-versa.

Sample run

```
Input the string: This Is A Test String.
The given sentence is: This Is A Test String.
After Case REVERSAL the string is: tHIS iS a tEST sTRING
```

P 26. Create a program to compute the mean and standard deviation of an array of data given below and displays the difference between each value and the mean.

15.3	4	90	34	2.5	104	7	25	82

For MAX_ITEM data items, if we assume that **x** is an array whose lowest subscript is 0, the standard deviation is given by the formula.

$$\text{sdandard deviation} = \sqrt{\frac{\sum_{i=0}^{\text{MAX\_ITEM-1}} \texttt{x[i]}^2}{\text{MAX\_ITEM}} - \text{mean}^2}$$

P 27. You have two independent sorted arrays of size m, and n respectively, where $m, n > 0$. You are required to merge the two arrays such that the merged array will be in sorted form and will contain exactly $m + n$ number of elements. You are not allowed to use any kind of sorting algorithm. Design your program to meet the above given requirement. Assume the elements of the array are non-negative integers. The elements can be read from the keyboard or can be generated randomly.

Example 1 :

First array:

12	20	24	32

Second array:

7	8	65	105

The merged sorted array:

7	8	12	20	24	32	65	105

Example 2 :

First array:

12	20	24

Second array:

7	8	65	105

The merged sorted array:

7	8	12	20	24	65	105

Example 3 :

First array :

12	20	24	100	120	130

Second array:

17	28	105	110

The merged sorted array:

12	17	20	24	100	105	110	120	130

P 28. Write a program to declare one array for storing the square roots of the integers from 0 through 10 and a second array for storing the cubes of the same integers.

P 29. Write a program to use the user-defined function **`multiply`** that takes two type **`int`** array input arguments and their effective size and produces a result array containing the sums of corresponding elements. For example, for the three-element input arrays 5 1 7 and 2 4 2 , the result would be an array containing 7 3 5 .

P 30. Write a program to copy the distinct elements of an int type array to another int type array. For example, if the input array is 4 7 7 3 2 5 5 then the output array will be 4 7 3 2 5.

P 31. Write a program which uses a user defined function, **`find_largest()`** to find the largest element of an integer array.

P 32. The **bubble sort** is another technique for sorting an array. A bubble sort compares adjacent array elements and exchanges their values if they are out of order. In this way, the smaller values ”bubble“ to the top of the array (toward element 0), while the larger values sink to the bottom of the array. After the first pass of a bubble sort, the last array element is in the correct position; after the second pass the last two elements are correct, and so on. Thus, after each pass, the unsorted portion of the array contains one less element. Write and test a function that implements this sorting method.

P 33. The *binary search* algorithm that follows may be used to search an array when the elements are in order. The algorithm for binary search given as;

1. Let **`bottom`** be the subscript of the initial array element.
2. Let **`top`** be the subscript of the last array element.
3. Let **`found`** be false.
4. Repeat as long as **`bottom`** isn't greater than **`top`** and the target has not been found
    5. Let **`middle`** be the subscript of the element halfway between **`bottom`** and **`top`**.
    6. if the element at **`middle`** is the target
        7. Set **`found`** to true and **`index`** to **`middle`**.

    else if the element at **`middle`** is larger than the target
        8. Let **`top`** be **`middle - 1`**.

    else
        9. Let **`bottom`** be **`middle + 1`**.

Write and test a function **`binary_srch`** that implements this algorithm for an array of integers. When there is a large number of array elements, which function do you think is faster: **`binary_srch`** or the linear search algorithm.

P 34. Implement the following algorithm for linear search that sets a flag (for loop control) when the element being tested matches the target.

1. Assume the target has not been found.
2. Start with the initial array element.
3. repeat while the target is not found and there are more array elements
    4. if the current element matches the target
        5. Set a flag to indicate that the target has been found.

    else
        6. Advance to the next array element.
7. if the target was found
        8. Return the target index as the search result.

    else
        9. Return -1 as the search result.

Create a user-defined function with prototype **`int linear_search(const int arr[ ], int target, int n);`** in your program to search the target element.