Practical 9

1. Write a program to print the value and memory address of an integer variable. Print the memory address in hexadecimal and decimal formats. Use dereference operator to print the value stored in the variable again.

2. Declare an integer variable and an integer pointer variable. Assign address of the integer variable to integer pointer variable. Print the content of the pointer variable.

3. Write a c program in the following steps.
    - Declare an integer variable and pointer variable.
    - Assign a value to the variable. Print the value and memory address of variable.
    - Assign the memory address of the variable to pointer variable. Print the content and address of the pointer variable. Output the value stored at the memory address pointed to by the pointer variable.
    - Assign a different value to the integer variable. Print the content and address of the pointer variable. Output the value stored at the memory address pointed to by the pointer variable.
    - Alter the value stored at the memory address indicated by the pointer variable. Print the address and value of the integer variable.

4. Calculate the square of a number. Declare an integer variable and assign a value. Calculate the square of the assigned value and store the output in the same variable you declared before. Print the output.
    i. Declare a variable in the main function and assign a value.
    ii. Write a separate function (calculateSquare) (return type should be void) to calculate the square of a number. Pass a pointer of integer type to the function as a parameter.
    iii. In the main function, call “calculateSquare” and pass the memory address of declared variable as a function argument.

5. Write a program to access elements of an array using pointers. Print the values and memory addresses of the locations.

6. Write a program to reverse a string using pointers.

- Declare an array type of char.
- Let the user input a string using declared array.
- Use “strlen” to get the length of the string.
- Pass the array as a pointer to the function.

7. Write a C function called `mergeSortedArraysInPlace` that takes two sorted arrays `arr1` and `arr2`, along with their respective sizes `size1` and `size2`, as input. Another empty array should also feed into the function to hold the merged array. The function should merge the two arrays in-place, that is, it should modify `arr3` to contain the merged result in sorted order. (Suppose that two arrays are already in a sorted condition.)

    - Declare two arrays and initialize in the main function.
    - Declare a third array to add merged values of the first two arrays.
    - Write a function to merge the arrays and add the merged values to the third array. Pass all the arrays as pointers to the function.

8. Write a C function called `removeDuplicates` that takes a sorted integer array `arr` and its size `size` as input. The function should remove duplicate elements from the array in-place, modify the original array to contain only unique elements, and modify the new size of the array without duplicates. The array and size of the array should be passed as pointer variables to the function.
   (Suppose that array is already in a sorted condition.)

9. Declare a student structure using struct keyword. Structure should include a string to student name and integer to store student age. Declare a variable of type structure student. Write a function to get the user input to student name and age.

10. Declare an array to store students declared in question 9. Store the student information in the array.