Tasks

1. **Pointers**
   - **a.** Declare variables `char a8`, `short a16`, `long a32`, `int i32, double d64`.
   - **b.** Declare pointers `char * pa8`, `short * pa16, long * pa32, int * pi32, double * pd64` and initialize them with the suitable addresses e.g. use part-a's variables.
   - **c.** Print sizes of all the variables declared in part-a and part-b using `sizeof()` instruction.

2. **Pointer Initializations**

   For the generic pointer `void * pv` as well as the pointer declarations of Task 1-b, check following;
   - **a.** `pa32 = NULL;`
   - **b.** `pa32 = 0x0;`
   - **c.** `pa32 = 1;`
   - **d.** `pa32 = &3;`
   - **e.** `pa32 = &(a32+3);`
   - **f.** `pa32 = pd64 ;`
   - **g.** `pv = pd64 ;`
   - **h.** `pa32 = pv = pd64 ;`
   - **i.** `*pv =234;`
   - **j.** `pa32 = (int *) pd64 ;`

3. **Pointer Arithmetic**

   Observe pointer variables in watch window after performing following operations on pointers of task 1-b;
   - **a.** `pa32++;     pa32 += 1; pa32 = pa32 + 16;     pa32 -= 3;`
   - **b.** `pa8++;      pa8 += 1;  pa8 = pa8 + 16;       pa8 -= 3;`

4. **Pointers & Arrays**
   - **a.** Declare `double ar[4] = { 1.0, 2, -3, 4}; and double * par;` and then check;
     - i. `par  = ar;   par  = &ar[2];`
     - ii. `ar[2] = par[0] + par[1];`
     - iii. `par[2] = ar[0] + *(ar+1);`
     - iv. `*par = ar[1] + ar[3];`
     - v. `*(par+2) = ar[1] + ar[3];`
     - vi. `*ar = par[1] + *(par+1);`
   - **b.** Declare `double * const pdc = &ar[2]; and` then check;
     - i. `pd++;`
     - ii. `pdc++;`
     - iii. `ar++;`
     - iv. `par++;`

5. **Function parameter passing: Arrays, Pointers**
   - **a.** Write a function **`swap_by_value()`** which swaps 2 variables. It's prototype would look like;
     `void swap_by_value(int a, int b)`
     This function also prints the swapped values.
     Moreover, also print the variables after returning from function.
   - **b.** Do the above task again, but now by using pointer variables.
     `void swap(int * const a, int * const b)`
   - **c.** If point input argument is simply input, then follow following technique.
     `int  dot_product(const int * const a, const int * const b)`

6. **Dynamic Memory Allocation**
   1. Use of **`malloc()`**, **`calloc()`** and **`free()`**.
   2. Freeing the already freed pointer.
   3. Try to allocate maximum possible size using **`malloc()`**.