1. - 1.1 True
   - 1.2 ...
   - 1.3 ...
   - 1.4 ...

2. - 2.1
     0xF9320904
     0xF93209AC
     0x2A
   - 2.2
     (a) calculate the sum of ~~all the elements~~(first n elements) in arr

     (b) sum = - (the number of zeros in arr)

     (c) ~~swap the value of x and y~~(Ultimately does not change the value of either x or y.)

     (d) `~(x ^ y)`
     also: `x & y | ~ (x | y)`
     x == y

3. 3.1

   (a)
   ```
   void swap(int *a, int *b) {
   int temp = *a;
   *a = *b;
   *b = temp;
   }
   ```

   (b)
   ```
   int mystrlen(char* a) {
   int ret = 0;
   while (*a != '\0') {
   ret++;
   a++;
   }
   return ret;
   }
   ```

   3.2

   (a) sizeof(summands) is incorrect, use an extra argument to pass the size of summands

   (b) Can't understand the question.

(c) if the memory of dst is less than src, it will cause an error.
multi suffix ++ (or prefix)should not be in one line, it's confusing and will easily cause trouble.

(d) line 5 is wrong: replaceptr is of type "char", not "char *"

4. 4.1

   (a) static

   (b) stack

   (c) static(maybe)Right!

   (d) static (Code, static, or stack)

   (e) code

   (f) heap

   (g) staticor stack

   4.2

   (a)
   ```
   int* arr = (int *) calloc (k, sizeof(int));
   ```

   (b)
   ```
   str = (char*) malloc ((p + 1) * sizeof(char))
   ```

   (c)
   ```
   int **a = (int**) malloc (n * sizeof(int*));
   for (int i = 0; i < n; ++i) {
   a[i] = (int*) calloc (m, sizeof(int));
   }
   ```

   4.3
   line 8 and line 9 both have problems.
   line 8: if the length of input is less than 11, important_stuff will be assigned rubbish value
   line 9: ”\0”$isactually'\backslash 0'and'\backslash 0'$, not a character.
   4.4

   ```
   void prepend(struct ll_node** lst. int value) {
   struct ll_node* st = (struct ll_node*) malloc (sizeof(struct ll_node));
   st -> next = *lst;
   st -> value = value;
   *ll_node = st;
   }
   ```

   reason for double * :omitted
   4.5

```
while (*lst != NULL) {
struct ll_node* temp = (struct ll_node*) malloc (sizeof(struct ll_node));
*lst = (*lst) -> next;
free(temp);
}
```