Ozyegin University CS 321 Programming Languages Sample Problems on Imperative Programming

1.	(PLC Ex. 7.2.(i)) Write a C program containing a function void arrsum(int n, int arr[], int *sump) that computes and returns the sum of the first n elements of the given array arr. The result must be returned through the sump pointer.
2.	(PLC Ex. 7.2.(ii)) Write a C program containing a function void squares(int n, int arr[]) that given n and an array arr of length n or more, fills $arr[i]$ with $i*i$ for $i = 0,, n-1$.
3.	Write a recursive C function void fib(int n, int *res) that computes the n th fibonacci number and returns it through the res pointer.

4. Assuming that the environment and the store are initially empty, give a possible environment and store at the end of the following piece of C program.

```
int n = 38;
int a[3] = {5, 9, 13};
int *p;
p = &a[1];
*p = n;
*(p+1) = a[0]*2;
p = &n;
p++;
```

5. In C++, parameters of functions that are declared using the & operator are passed by reference. What is the output of the C++-like program below?

```
void main() {
  int a[3] = {3, 7, 10};
  int m = 56;
  int n = 99;
  mystery(m, n, a);
  print m, n, a[0], a[1], a[2]
}
void mystery(int x, int &y, int a[]) {
  a[0] += 1;
  int temp = x;
  x = y;
  y = temp;
}
```

6. An array can be represented as a pointer. For instance, the array definition

int
$$a[4] = \{12, 13, 14, 15\};$$

can be represented using the following env/store:

Explain how you can use this representation to find the length of an array.

7. A C-like program is given below.

Starting from the env. and store given below, show the environment and the store after **each** statement.

8. In C++, parameters of functions that are declared using the & operator are passed by reference. A C++ program is given below.

```
void main() {
                               void f(int &x, int *y, int z[]) {
 int a[3] = {8, 15, 6};
                                 // B
 int b = 88;
                                 *y = *y + 1;
 int c = 33;
                                 // C
                                 y = &(z[1]);
  // A
 f(b, &c, a);
                                 // D
  // I
                                 y++; x++;
                                 // E
                                 *y = x + 10;
                                 x = *y;
                                 // G
                                z[2] = -1;
                                 // H
```

The *environment* maps names to locations; the *store* maps locations to values. Suppose the environment and the store at point A are as given below. Give possible environment and store configurations for points B–I. It is OK to show the changes only.

A:	61 61
C: 55 S 13 O 30 86 55	
B: y: 50 51 52 53 54 55 56 57 58 59 60	
B: y:	
B: y:	
z:	61
	61
x: 50 51 52 53 54 55 56 57 58 59 60	n i
C: y: 50 51 52 53 54 55 56 57 58 59 60	
z:	
x: 50 51 52 53 54 55 56 57 58 59 60	61
D: y:	
Z:	
x: 50 51 52 53 54 55 56 57 58 59 60	61
E: y:	
Z:	
x: 50 51 52 53 54 55 56 57 58 59 60	61
F: y: 50 51 52 53 54 55 56 57 58 59 60	101
z:	
x: 50 51 52 53 54 55 56 57 58 59 60	61
G: y:	
Z:	
x: 50 51 52 53 54 55 56 57 58 59 60	61
H: y:	
z:	
a: 50 51 52 53 54 55 56 57 58 59 60	61
I: b: 50 51 52 55 54 55 56 57 58 59 60	101
c:	