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.

Solution:

```c
void arrsum(int n, int arr[], int *sump) {
  int sum = 0;
  for (int i = 0; i < n; i++) {
    sum += arr[i];
  }
  *sump = sum;
}
```

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` $= 0, \ldots, n-1$.

Solution:

```c
void squares(int n, int arr[]) {
  for (int i = 0; i < n; i++) {
    arr[i] = i * i;
  }
}
```

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.

Solution:

```c
void fib(int n, int *res) {
  if (n == 0 || n == 1) {
    *res = 1;
  } else {
    int f1;
    fib(n-1, &f1);
    int f2;
    fib(n-2, &f2);
    *res = f1 + f2;
  }
}
```

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++;
```

Solution:

```
* ENV
  n: 70
  a: 74
  p: 75

* STORE
  ---------------------------------------
    | 38 |  5 | 38 | 10 | 71 | 71 |
  ---------------------------------------
      70   71   72   73   74   75
```

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;
}
```

Solution: 56 56 4 7 10

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:

```
Env =>      a: 54

Store =>    -------------------------------------
               | 12 | 13 | 14 | 15 | 50 |    |
            -------------------------------------
                 50   51   52   53   54   55
```

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

Solution: Find the difference between the address and the value of the array variable. That is, `&a - a`

7. A C-like program is given below.

```
m = &n;          // A
*m = k[2];       // B
k--; m++;        // C
*k = *m;         // D
m[2] = n;        // E
```

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

```
Env =>      m: 50, n: 51, k:56

Store =>    ----------------------------------------
               | 48 | 53 | 49 | 17 | 41 | 50 | 52 |
            ----------------------------------------
                 50   51   52   53   54   55   56
```

Solution: After A:

```
Env =>      m: 50, n: 51, k:56

Store =>    ----------------------------------------
               | 51 | 53 | 49 | 17 | 41 | 50 | 52 |
            ----------------------------------------
                 50   51   52   53   54   55   56
```

After B:

```
Env =>      m: 50, n: 51, k:56

Store =>    ----------------------------------------
               | 51 | 41 | 49 | 17 | 41 | 50 | 52 |
            ----------------------------------------
                 50   51   52   53   54   55   56
```

After C:

```
Env =>      m: 50, n: 51, k:56

Store =>    ----------------------------------------
               | 52 | 41 | 49 | 17 | 41 | 50 | 51 |
            ----------------------------------------
                 50   51   52   53   54   55   56
```

After D:

```
Env =>      m: 50, n: 51, k:56

Store =>    ----------------------------------------
               | 52 | 49 | 49 | 17 | 41 | 50 | 51 |
            ----------------------------------------
                 50   51   52   53   54   55   56
```

After E:

```
Env =>      m: 50, n: 51, k:56

Store =>    ----------------------------------------
               | 52 | 49 | 49 | 17 | 49 | 50 | 51 |
            ----------------------------------------
                 50   51   52   53   54   55   56
```

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

```
void main() {
  int a[3] = {8, 15, 6};
  int b = 88;
  int c = 33;
  // A
  f(b, &c, a);
  // I
}
```

```
void f(int &x, int *y, int z[]) {
  // B
  *y = *y + 1;
  // C
  y = &(z[1]);
  // D
  y++; x++;
  // E
  *y = x + 10;
  // F
  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:

Env	
a:	53
b:	54
c:	55

Store

50	51	52	53	54	55	56	57	58	59	60	61
8	15	6	50	88	33						

B:

Env	
x:	54
y:	56
z:	57

50	51	52	53	54	55	56	57	58	59	60	61
						55	50				

C:

Env	
x:	
y:	
z:	

50	51	52	53	54	55	56	57	58	59	60	61
					34						

D:

Env	
x:	
y:	
z:	

50	51	52	53	54	55	56	57	58	59	60	61
						51					

E:

Env	
x:	
y:	
z:	

50	51	52	53	54	55	56	57	58	59	60	61
				89		52					

F:

Env	
x:	
y:	
z:	

50	51	52	53	54	55	56	57	58	59	60	61
		99									

G:

Env	
x:	
y:	
z:	

50	51	52	53	54	55	56	57	58	59	60	61
				99							

H:

Env	
x:	
y:	
z:	

50	51	52	53	54	55	56	57	58	59	60	61
		-1									

I:

Env	
a:	53
b:	54
c:	55

50	51	52	53	54	55	56	57	58	59	60	61