

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, \dots, 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:

```
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.

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
```

