

**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:**

```
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, ..., n - 1`.

**Solution:**

```
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:**

```
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:** After the third line, that is, after the declarations are handled, we may have an environment and a store such as below, where the value of `p` is garbage (i.e. it's not set, therefore it could be anything).

Env	Store						
n: 70	70	71	72	73	74	75	
a: 74		38	5	9	13	71	XX
p: 75							

Executing the statement `p = &a[1];` gives us

n:	70	70	71	72	73	74	75
a:	74	38	5	9	13	71	<b>72</b>
p:	75						

Executing the statement `*p = n;` gives us

n:	70							
a:	74							
p:	75							

	70	71	72	73	74	75	
	38	5	<b>38</b>	13	71	72	

Executing the statement `*(p+1) = a[0]*2;` gives us

n:	70							
a:	74							
p:	75	70	71	72	73	74	75	
		38	5	38	<b>10</b>	71	72	

Executing the statement `p = &n;` gives us

n: 70	70	71	72	73	74	75	
a: 74	38	5	38	10	71	<b>70</b>	
p: 75							

Executing the statement `p++;` gives us

n:	70							
a:	74							
p:	75	70	71	72	73	74	75	
		38	5	38	10	71	<b>71</b>	

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

```

