NetId: iv447
University Id: N17385760

Chapter-9

Problem set: 5

Consider the following program written in C syntax:

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

Void main() {
  int value = 2, list[5] = {1, 3, 5, 7, 9};
   swap (value, list[0]);
   swap (list[0], list[1]);
    swap (value, list[value]);
  }
```

For each of the following parameter-passing methods, what are all of the value of the variables value and list after each of the three calls to swap?

(a) Passed by value
(b) passed by reference
(c) passed by value-result

(a) Passed by value

Solution: The pass-by-value copies the values of the actual parameters in the formal parameters. The changes made in the formal parameters will not change the actual parameters.

Therefore, the values of the variables and list will remain the same after the function calls. The values will be as follows:

list [ ] = { 1, 3, 5, 7, 9 }
Value = 2

(b) Passed by reference:-

The pass-by-reference function call accepts the address of the actual parameters. Therefore, changes made in the formal parameters will reflect in the actual parameters since the operations are performed using the addresses.

Changes will occur in following sequence:

Initially, Value = 2
list[ ] = { 1, 3, 5, 7, 9 }

- The first function call swaps the values of the parameters value and list [0] resulting in following values:

Value = 1
list[ ] = { 2, 3, 5, 7, 9 }

- The second function call swaps the value of the parameters list[0] and list[1] resulting in the following values:

Value = 1
list[] = {3, 2, 5, 7, 9}

- The third function call swaps the values of the parameters value and list[value] resulting in the following values:

Value = 2
list[] = {3, 1, 5, 7, 9}

(c) Passed by Value Result:

The pass-by-value-result copies the values of the actual parameters in the formal parameters and after the function ends, the modified values are copied back in the actual parameters.

The change in the values of the variables value and list:

Initially
Value = 2
list[] = {2, 3, 5, 7, 9}

- The first function call swaps the values of the parameters value and list[0] resulting in the

following values:

Value = 1
list[] = {2, 3, 5, 7, 9}

- The second function calls swaps the values of the parameters list[0] and list[1] resulting in the following values:

Value = 1
list[] = {3, 2, 5, 7, 9}

- The third function call swaps the values of the parameters Value and list[Value] resulting in the following values:

Value = 2
list[] = {3, 1, 5, 7, 9}

Problem Set 7:
Consider the following program written in C Syntax:

```
Void fun (int first, int second) {
    first += first
    second += second
    }
Void main () {
```

```
int list[2] = {1, 3};
fun(list[0], list[1]);
}
```

For each of the following parameter - passing methods, what are the values of the list array after execution?

(a) Passed by Value

When parameters are passed by Value to a Subprogram, Subprograms makes a copy of the parameters and any changes to the formal parameters do not reflect to the actual parameter.

The Value of List array = {1, 3}

(b) Pass by reference:

When parameters are passed by reference to a Subprogram, an access path of the actual parameter is provided to the formal parameter. Now formal parameters have access to the actual parameter. Any changes to the formal parameters are reflected to the actual parameter.

The value of list array after parameters are passed by reference is = {2, 6}

© Pass by Value-Result :

Sol: When parameters are passed by Value-Result to a Subprogram, Subprogram copies the value of actual parameter to the formal parameter. When the Subprogram completes its execution, the values of formal parameters are copied again to the actual Parameters. The output of pass by Value-result is same as when parameters are passed by Reference.

The value of list array when parameters are passed by Value-result is = {2, 6}