COM2002 INTERMEDIATE PROGRAMMING
2024 – 2025 SPRING

Laboratory Week: 10 – 14 February 2025

Topic : Pointers

Program-1 : Selection Sort (selection-sort.c)

Definition : The program sorts the elements of array in either ascending or descending order by using selection sort technique.

This program has several different tasks:

- ✓ **Sorting elements by using selection sort:** Selection sort algorithm divides the list into two parts:
  - the sorted part at the left end that is empty.
  - the unsorted part at the right end that is the entire list.

  In main program, the user selects sorting order either ascending or descending. According to sorting order, (1) the smallest (minimum) element is selected for ascending order (2) the biggest (maximum) element is selected for descending order, from unsorted array and swapped with the leftmost element. That element becomes a part of the sorted array. In every iteration of algorithm, either the minimum element (considering ascending order) or the maximum element (considering descending order) from the unsorted subarray is moved to the sorted subarray.

  Assume that the following array contains integer elements where LENGTH is the size of array.
  int my_array[LENGTH] = {12, 36, 27, 8, 77, 43, 22, 58, 60, 85};
  Sort the given array by using "**void selection_sort(int array[], char s_order)**" function.
- ✓ **Swapping elements:** Swap the first element with the second element by using "**void swap(int *first, int *second)**" function.
- ✓ **Printing array:** Print the given array by using "**void display_array(int array[])**" function.

Expected output-1:

```
The array before sort:
12   36   27   8   77   43   22   58   60   85
Choose the sorting order (either ascending or descending) (A/D) : A
The array after selection sort:
8   12   22   27   36   43   58   60   77   85
```

Expected output-2:

```
The array before sort:
12   36   27   8   77   43   22   58   60   85
Choose the sorting order (either ascending or descending) (A/D) : D
The array after selection sort:
85   77   60   58   43   36   27   22   12   8
```

Program-2 : Dollar (dollar.c)

Definition : The program shows how to pay that amount using the smallest number of $20, $10, $5, and $1 bills. Prompt the user to enter U.S. dollar amount.

The program includes the following function:

void pay_amount(int dollars, int *twenties, int *tens, int *fives, int *ones);

The function determines the smallest number of $20, $10, $5, and $1 bills necessary to pay the amount represented by the *dollars* parameter. The *twenties* parameter points to a variable in which the function will store the number of $20 bills required. The *tens*, *fives*, and *ones* parameters are similar.

> **Hint:** Divide the amount by 20 to determine the number of $20 bills needed, and then reduce the amount by the total value of the $20 bills. Repeat for the other bill sizes. Be sure to use integer values throughout, not floating-point numbers.

Expected output:

```
Enter a dollar amount: 93

$20 bills: 4
$10 bills: 1
$5 bills: 0
$1 bills: 3
```

Program-3 : What is the output of the following program?

```
#include <stdio.h>
int value4 = 1;

int function(int* param1, int param2, int* param3);

int main(void)
{
    int value1 = 7, value2 = 3, value3 = 2;
    printf("Print in main: %d %d %d %d \n", value1, value2, value3,
value4);
    value4 = function(&value1, value2, &value3);
    printf("Print in main: %d %d %d %d \n", value1, value2, value3,
value4);
    value4 = function(&value1, value2, &value3);
    printf("Print in main: %d %d %d %d \n", value1, value2, value3,
value4);

    return 0;
}
int function(int* param1, int param2, int* param3)
{
    int static value6 = 11;

    *param1 = *param1 + 2;
    param2 = param2 + 5;
```

```
    ++*param3;
    param3 = param1;
    value6 = value6 + *param3;
    printf("Print in function: %d %d %d %d %d \n", value4, *param1,
param2, *param3, value6);
    param3 = &param2;

    return *param3;
}
```