Preprocessor directive Lab Assignments

1) Write down a macro to find out the biggest of two numbers.

MAX(x,y) should define code to find biggest of x and y. Implement array sorting function using this macro (Bubble sort)

```
#include <stdio.h>
#define MAX(x, y) ((x) > (y) ? (x) : (y))

void bubbleSort(int arr[], int n) {
   for (int i = 0; i < n - 1; i++) {
      for (int j = 0; j < n - 1 - i; j++) {
         if (MAX(arr[j], arr[j + 1]) == arr[j + 1]) {
            int temp = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = temp;
         }
      }
   }
}

int main() {
   int arr[] = {4, 2, 8, 1, 5};
   int n = 5;
   bubbleSort(arr, n);
   for (int i = 0; i < n; i++) {
     printf("%d ", arr[i])
```

```
  }
   return 0;
}
```

Output: 8 5 4 2 1

2) Write down a macro to find the biggest of four numbers using biggest of 2 macro.

```
#include <stdio.h>

#define MAX(x, y) ((x) > (y) ? (x) : (y))
#define MAX4(a, b, c, d) MAX(MAX(a, b), MAX(c, d))

int main() {
   int a = 12, b = 25, c = 7, d = 19;
   int biggest = MAX4(a, b, c, d);
   printf("The biggest number is: %d\n", biggest);
   return 0;
}
```

Output: The biggest number is: 25

3) WAP to print if a character is alphanumeric or special character, using macro

Conditions

```
#include <stdio.h>

#define IS_UPPER(c) ((c) >= 'A' && (c) <= 'Z')
#define IS_LOWER(c) ((c) >= 'a' && (c) <= 'z')
#define IS_DIGIT(c) ((c) >= '0' && (c) <= '9')
#define IS_ALNUM(c) (IS_UPPER(c) || IS_LOWER(c) || IS_DIGIT(c))
```

```
int main() {
   char ch;
   printf("Enter a character: ");
   scanf(" %c", &ch);

   if (IS_ALNUM(ch))
      printf("Alphanumeric character\n");
   else
      printf("Special character\n");

   return 0;
}
```

Output: Enter a character: A

Alphanumeric character

Enter a character: #

Special character

4) Define a macro that receives an array and the number of elements in the array as arguments. Write a program using this macro to print out the elements of the array. Try using this macro for different data types of arrays.

```
#include <stdio.h>

#define PRINT_INT_ARRAY(arr, n) for (int i = 0; i < n; i++) printf("%d ", arr[i]); printf("\n");
#define PRINT_FLOAT_ARRAY(arr, n) for (int i = 0; i < n; i++) printf("%.1f ", arr[i]); printf("\n");
#define PRINT_CHAR_ARRAY(arr, n) for (int i = 0; i < n; i++) printf("%c ", arr[i]); printf("\n");
```

```
int main() {
    int a[] = {1, 2, 3};
    float b[] = {1.1, 2.2, 3.3};
    char c[] = {'A', 'B', 'C'};

    PRINT_INT_ARRAY(a, 3);
    PRINT_FLOAT_ARRAY(b, 3);
    PRINT_CHAR_ARRAY(c, 3);

    return 0;
}
```

Output: 1 2 3

1.1 2.2 3.3

A B C

5) Define a generic function, for different types of array printing , by taking array and array size as arguments

```
#include <stdio.h>

void printIntArray(int *arr, int n) {
    for (int i = 0; i < n; i++) printf("%d ", arr[i]);
    printf("\n");
}

void printFloatArray(float *arr, int n) {
```

```
    for (int i = 0; i < n; i++) printf("%.1f ", arr[i]);
    printf("\n");
}

void printCharArray(char *arr, int n) {
    for (int i = 0; i < n; i++) printf("%c ", arr[i]);
    printf("\n");
}

#define printArray(arr, n) _Generic((arr), \
    int*: printIntArray, \
    float*: printFloatArray, \
    char*: printCharArray \
)(arr, n)

int main() {
    int a[] = {1, 2, 3};
    float b[] = {1.1, 2.2, 3.3};
    char c[] = {'A', 'B', 'C'};

    printArray(a, 3);
    printArray(b, 3);
    printArray(c, 3);

    return 0;
}
```

Output: 1 2 3

1.1 2.2 3.3

A B C

6) define a macro to generate swapping function for int, float , double and character datatypes, as a generic function using macros.

```
#include <stdio.h>

#define DEFINE_SWAP(type) void swap_##type(type *a, type *b) { type t = *a; *a = *b; *b = t; }

DEFINE_SWAP(int)
DEFINE_SWAP(float)
DEFINE_SWAP(double)
DEFINE_SWAP(char)

int main() {
    int a = 1, b = 2;
    float x = 1.5, y = 2.5;
    double m = 3.3, n = 4.4;
    char c1 = 'X', c2 = 'Y';

    swap_int(&a, &b);
    swap_float(&x, &y);
    swap_double(&m, &n);
    swap_char(&c1, &c2);

    printf("%d %d\n", a, b);
```

```
    printf("%.1f %.1f\n", x, y);
    printf("%.1lf %.1lf\n", m, n);
    printf("%c %c\n", c1, c2);

    return 0;
}
```

Output:2 1

2.5 1.5

4.4 3.3

Y X