Exercise 1. Does the following C program compile? Motivate your answer if you think it does not.

```c
#include<stdio.h>

#define E 2.73
#ifdef PI
#define PI 3.14
#endif

int main()
{
    printf("E is \%.2f\n", E);
    printf("PI is \%.2f\n", PI);
    return 0;
}
```

Exercise 2. Does the following C program compile? Motivate your answer if you think it does not.

```c
#include<stdio.h>

#include<stdio.h>

#define E 2.73
#ifdef QQ
#define PI PI
#endif

int main()
{
    printf("E is \%.2f\n", E);
    return 0;
}
```
Exercise 3. Write the output generated by the following C program.

```c
#include<stdio.h>

#define SWAP(T,x,y) { T t=x;x=y;y=t;}

int main()
{
    int s=1;
    int t=0;

    SWAP(int,t,s);
    SWAP(int,s,t); // watch out

    printf("s = %d\n",s);
    printf("t = %d\n",t);
    return 0;
}
```

Exercise 4. Refactor the following C program using appropriate macros to avoid duplication of code patterns. (Use tricks similar to the `foreach` macro discussed in the lecture.)

```c
int main() {
    float f1[10];
    float f2[10];
    double d1[20];
    double d2[20];
    for (int i = 0; i < 10; i++)
        f1[i] = f1[i] * f1[i];
    for (int i = 0; i < 10; i++)
        f2[i] = f2[i] + f2[i];
    for (int i = 0; i < 20; i++)
        d1[i] = d1[i] * d1[i];
    for (int i = 0; i < 20; i++)
        d2[i] = d2[i] + d2[i];
}
```

Exercise 5. A possible way to avoiding variable capture is to use temporary variables with macro-specific names. However, this may not always work. Explain why the following macro definition does not guarantee referential integrity.

```c
#define SWAP(T,x,y) T __SWAP_temp = y; y = x; x = __SWAP_temp;
```

Exercise 6. Does the following C program compile? Motivate your answer if you think it does not.

```c
#include<stdio.h>

#define FACT(N) (N > 1) ? (N*FACT(N-1)) : 1
```
int main() {
    int k = FACT(5);
    printf("%d",k);
}

Exercise 7. The modulus of a real number is its nonnegative value obtained disregarding its sign. The macro MODULUS in the following program attempts this computation.
#include<stdio.h>
#define x 0,-1
#define MODULUS(a) ((a)<0?-a:a)

int main()
{
    printf("%d\n", MODULUS(x));
    return 0;
}

Write the output of the above program. Hint: make sure you understand the “comma” operator ‘,’ in C.