Домашна работа - Седмица №5

Задача 1: Препроцесорни макроси

1.1 Направете макрос **MAX(x, y, z)** , който връща най-голямото от три числа.
1.2 Направете макрос **MIN(x, y, z)** , който връща най-малкото от три числа.
1.3 Направете побитов макрос **SETBIT(mask, bit)** , който установява определен бит в **1** в побитовата маска.
1.4 Направете побитов макрос **CLEARBIT(mask, bit)** , който установява определен бит в **0** в побитовата маска.
1.5 Направете побитов макрос **INVERSEBIT(mask, bit)** , който обръща определен в побитовата маска.
1.6 Направете побитов макрос **CHECKBIT(mask, bit)**, който връща 0 или 1 в зависимост от състоянието на бита.
1.7 Направете побитов макрос **SWAP(a, b)** , който разменя две променливи.

Задача 2: Етапи на компилиране на C програми

Създайте дадените по-долу **C** файлове. Компилирайте и изпълнете програмата с **GCC** като компилирате отделно двата файла стъпка по стъпка и след това ги свържете в изпълним файл **program.**

Използвате текстови редактори за *.i и *.s файловете, както и hex редактор (напр. **ghex**) за да ги отворите и анализирате. Използвайте програмата **objdump** и **readelf** за да прочетете обектните и ELF файла. Разгледайте различни опции (напр. **man readelf**).

File 1: program.c

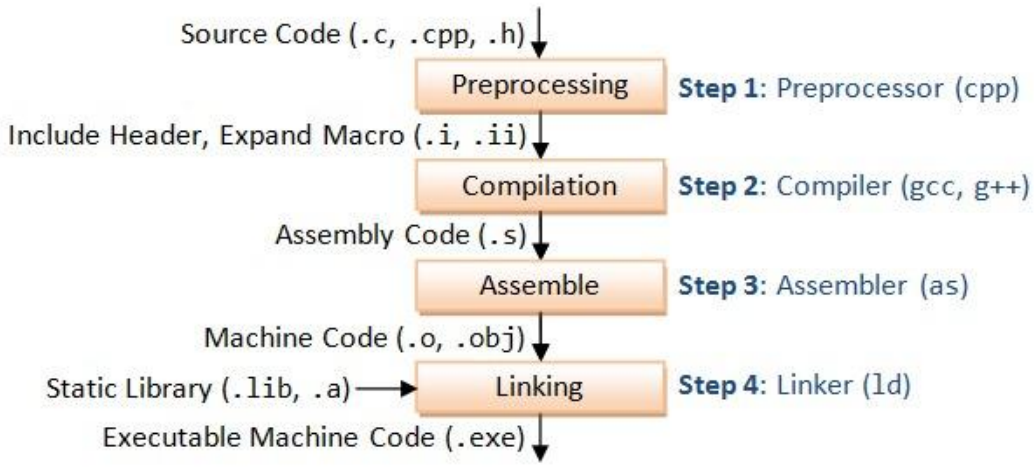
```
#include <stdio.h>

void swap(int*, int*);

int main(void)
{
    int num1, num2;
    printf("Enter first number: ");
    scanf("%d", &num1);
    printf("Enter second number: ");
    scanf("%d", &num2);
    printf("Before swap: Num1 = %d, Num2 = %d\n", num1, num2);
    swap(&num1, &num2);
    printf("After swap: Num1 = %d, Num2 = %d\n", num1, num2);
    return 0;
}
```

File 2: swap.c

```
void swap(int* num1, int* num2)
{
    *num1 ^= *num2;
    *num2 ^= *num1;
    *num1 ^= *num2;
}
```

Hint: Example for two file manual compilation and linking:

Step	File 1	File 2
Preprocessing	gcc -E file1.c -o file1.i	gcc -E file2.c -o file2.i
Compilation	gcc -S file1.i -o file1.s	gcc -S file2.i -o file2.s
Assemble	gcc -c file1.s -o file1.o	gcc -c file2.s -o file2.o
Linking	gcc file1.o file2.o -o program	



Направете компилацията като създадете хедърен файл: **swap.h**. Създайте необходимите *include guards*.

Задача 3: Динамична памет (Heap)

Направете програма, в която потребителят въвежда от стандартния вход цяло число **n**. След това да се създаде масив от реални числа от **n** на брой елементи в динамичната памет. Елементите на масива да се инициализират с произволни стойности между 0,0 и 1.0. След това се въвежда ново число **m** и масивът се да се разшири с нови **m** елемента. Новите елементи да се инициализират с числа между 1.0 и 2.0. След това да се въведе цяло число **p**. Масивът да се разшири с нови **p** елемента. Новите елементи да се инициализират с числа между 2.0 и 3.0. Накрая да се изведе масива в стандартния изход. Да се освободи заетата памет.

Да се следи дали всяка функция успешно заделя памет.

https://www.geeksforgeeks.org/dynamic-memory-allocation-in-c-using-malloc-calloc-free-and-realloc/

Бонус задачи:

*Задача 4: Бонус задача

Направете функция, която изчислява корен квадратен, без да използвате math.h или друга външна библиотека.

*

Задача 5: Бонус задача

Направете функция, която изчислява **sin(x)**, без да използвате math.h или друга външна библиотека.