PRÁCTICA 2
PROGRAMACIÓN 2

1. Describir lo que imprime el siguiente fragmento de código:

```
int *p, a = 4; b = 5;
p = &b;
*p *= 2;
printf("b=%d *p=%d\n", b, *p);          _____
printf("&b=%p p=%p &p=%p\n", &b, p, &p); _____
b = *p *3;
printf("b=%d *p=%d\n", b, *p);          _____
printf("&b=%p p=%p\n", &b, p);          _____
a = b;
p = &a;
(*p)++;
printf("b=%d a=%d *p=%d\n", b, a, *p);  _____
printf("&b=%p &a=%p p=%p &p=%p\n", &b, &a, p, &p); _____
                                        _____
```

- b=10 *p=10
- imprimirá las direcciones de memoria
- b=30 *p=30
- imprimirá las direcciones de memoria
- b=30 a=31 *p=31
- imprimirá las direcciones de memoria

2. Corregir el siguiente programa para que los valores de las variables a y b resulten ordenados de manera ascendente:

```
#include <stdio.h>
  int main(){
    int a = 30, b = 20;
    ordenadas(a, b);
    printf(" valor de a %d\tvalor de b %d\n", &a, &b);
    return 0;
  }
  void ordenadas(int x, int y){
    int* aux;
    if(x > y) {
      aux = x;
      x = y;
      y = aux;
    }
  }
```

```
#include <stdio.h>

int main(){
 int a = 30, b = 20;
 ordenadas(&a, &b);
 printf(" valor de a %d\tvalor de b %d\n", &a, &b);
```

```
 return 0;
}
void ordenadas(int *x, int *y){
 int aux;
 if(*x > *y) {
 aux = *x;
 *x = *y;
 *y = aux;
 }
}
```

3. Implementar un programa que cree dinámicamente 3 variables enteras, muestre su suma y su producto. Asegurarse de administrar correctamente la memoria e implementar una función para evitar duplicaciones de código en la creación y lectura de cada variable.

```
#include <stdio.h>
int main(){
 int suma,producto,*a, *b, *c;
 a = (int*)malloz(sizeof(int));
 b = (int*)malloz(sizeof(int));
 c = (int*)malloz(sizeof(int));
 printf("ingrese a: ");
 scanf("%d",a);
 printf("ingrese b: ");
 scanf("%d",b);
 printf("ingrese c: ");
 scanf("%d",c);
 suma = *a + *b + *c;
 producto = (*a)*(*b)*(*c);
 printf("suma: %d\tproducto: %d",suma,producto);
 free(a);free(b);free(c);
}
```

4. Desarrollar un programa que cree dinámicamente un arreglo de números reales que contenga N elementos (N es ingresado por teclado). Ingresar sus elementos y mostrar aquellos que sean positivos utilizando aritmética de punteros. Al finalizar, liberar la memoria solicitada en tiempo de ejecución.

```
#include <stdio.h>

int main(){
    int i,n;
```

```
    float *V;
    printf("ingrese n: ");
    scanf("%d",&n);
    V = (float)malloc(n * sizeof(float));

    //cargo vector
    for ( i = 0; i < n; i++) {
        printf("ingrese un elemento: ");
        scanf("%f",(V + i));
    }
    //muestro vector
    for (i = 0; i < n; i++) {
        if (*(V + i) > 0){
            printf("%.2f\n",*(V + i));
        }
    }

    free(V);
    return 0;
}
```

5. Desarrollar un programa que cree un arreglo estático de punteros a enteros, y luego cargue en él una cantidad desconocida de enteros (se encuentran en un archivo de texto). Posteriormente a la carga, mostrar aquellos que sean positivos. Luego, para finalizar, liberar la memoria solicitada en tiempo de ejecución.

```
#include <stdio.h>
#define MAX 30
int main(){
    int i, n = 0,*V[MAX];
    FILE *arch;
    arch = fopen("numeros.txt","r");
    //primero cargo el vector
    while (!feof(arch)){
        V[n] = (int*)malloc(sizeof(int));
        fscanf(arch,"%d",V[n]);
        n++;
    }
    fclose(arch);

    //muestro positivos

    for (i = 0;i < n; i++){
        if (*(V[i]) > 0){
            printf("%d\n",*V[i]);
        }
    }
    for (i = 0;i < n; i++){
```

```
        free(V[i]);
    }
    return 0;
}
```