POINTERS

1. Print address, value, and size of a variable

```
#include <stdio.h>
int main() {
   int num = 10;
   int *ptr;
   // Assign address of variable to pointer
   ptr = &num;
   printf("Value of variable: %d\n", num);
   printf("Address of variable: %p\n", &num);
   printf("Size of variable: %zu bytes\n", sizeof(num));
   printf("Value of pointer: %p\n", ptr);
   printf("Size of pointer: %zu bytes\n", sizeof(ptr));
   printf("Value pointed by pointer: %d\n", *ptr);
   return 0;
}
```

2. Add two variables using pointers

```
#include <stdio.h>
int main() {
   int num1, num2, sum;
   int *ptr1, *ptr2;
   printf("Enter two numbers: ");
   scanf("%d %d", &num1, &num2);
   // Assign addresses to pointers
   ptr1 = &num1;
   ptr2 = &num2;
```

```
    // Add using pointers
    sum = *ptr1 + *ptr2;
    printf("Sum: %d\n", sum);
    return 0;
}
```

3. Compare ASCII values of characters using pointers

```
#include <stdio.h>
int main() {
    char char1, char2;
    char *ptr1, *ptr2;
    // Assign pointers
    ptr1 = &char1;
    ptr2 = &char2;
    // Take input using pointers
    printf("Enter first character: ");
    scanf(" %c", ptr1);
    printf("Enter second character: ");
    scanf(" %c", ptr2);
    // Compare ASCII values
    if(*ptr1 > *ptr2) {
        printf("'%c' has higher ASCII value: %d\n", *ptr1, *ptr1);
    } else if(*ptr2 > *ptr1) {
        printf("'%c' has higher ASCII value: %d\n", *ptr2, *ptr2);
    } else {
        printf("Both characters have the same ASCII value: %d\n", *ptr1);
    }
```

```
    return 0;
}
```

4. Use one pointer to access different variables

```
#include <stdio.h>
int main() {
    int var1 = 10, var2 = 20, var3 = 30;
    int *ptr;
    // Assign address of var1 to pointer
    ptr = &var1;
    printf("Value of var1 using pointer: %d\n", *ptr);
    // Assign address of var2 to pointer
    ptr = &var2;
    printf("Value of var2 using pointer: %d\n", *ptr);
    // Assign address of var3 to pointer
    ptr = &var3;
    printf("Value of var3 using pointer: %d\n", *ptr);
    return 0;
}
```

5. Multiple pointers to same variable

```
#include <stdio.h>
int main() {
    int value = 100;
    int *ptr1, *ptr2, *ptr3;
    // Assign address of variable to all pointers
    ptr1 = &value;
    ptr2 = &value;
```

```
ptr3 = &value;
// Print initial values
printf("Original value: %d\n", value);
printf("Value using ptr1: %d\n", *ptr1);
printf("Value using ptr2: %d\n", *ptr2);
printf("Value using ptr3: %d\n", *ptr3);
// Change value directly
value = 200;
printf("\nAfter changing value directly to 200:\n");
printf("Value using ptr1: %d\n", *ptr1);
printf("Value using ptr2: %d\n", *ptr2);
printf("Value using ptr3: %d\n", *ptr3);

// Change using ptr1
*ptr1 = 300;
printf("\nAfter changing value using ptr1 to 300:\n");
printf("Original value: %d\n", value);
printf("Value using ptr1: %d\n", *ptr1);
printf("Value using ptr2: %d\n", *ptr2);
printf("Value using ptr3: %d\n", *ptr3);

// Change using ptr2
*ptr2 = 400;
printf("\nAfter changing value using ptr2 to 400:\n");
printf("Original value: %d\n", value);
printf("Value using ptr1: %d\n", *ptr1);
```

```
    printf("Value using ptr2: %d\n", *ptr2);
    printf("Value using ptr3: %d\n", *ptr3);

    // Change using ptr3
    *ptr3 = 500;
    printf("\nAfter changing value using ptr3 to 500:\n");
    printf("Original value: %d\n", value);
    printf("Value using ptr1: %d\n", *ptr1);
    printf("Value using ptr2: %d\n", *ptr2);
    printf("Value using ptr3: %d\n", *ptr3);
    return 0;
}
```

6. Size of different data type pointers

```
#include <stdio.h>
int main() {
    int *int_ptr;
    char *char_ptr;
    float *float_ptr;
    double *double_ptr;
    printf("Size of int pointer: %zu bytes\n", sizeof(int_ptr));
    printf("Size of char pointer: %zu bytes\n", sizeof(char_ptr));
    printf("Size of float pointer: %zu bytes\n", sizeof(float_ptr));
    printf("Size of double pointer: %zu bytes\n", sizeof(double_ptr));
    printf("\nAll pointers have the same size because they store memory addresses, ");
    printf("regardless of the data type they point to. The size of a pointer depends ");
    printf("on the architecture (32-bit or 64-bit) of the system.\n");
```

```
    return 0;
}
```

7. Find biggest of three numbers using pointers

```
#include <stdio.h>
int main() {
    int num1, num2, num3, largest;
    int *ptr1, *ptr2, *ptr3;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);
    // Assign addresses to pointers
    ptr1 = &num1;
    ptr2 = &num2;
    ptr3 = &num3;

    // Find largest using pointers
    if(*ptr1 >= *ptr2 && *ptr1 >= *ptr3) {
        largest = *ptr1;
    } else if(*ptr2 >= *ptr1 && *ptr2 >= *ptr3) {
        largest = *ptr2;
    } else {
        largest = *ptr3;
    }
    printf("The largest number is: %d\n", largest);
    return 0;
}
```

8. Rotate values of x, y, z using pointers

```
#include <stdio.h>
int main() {
   int x, y, z, temp;
   int *px, *py, *pz;
   printf("Enter three integers (x, y, z): ");
   scanf("%d %d %d", &x, &y, &z);
   printf("Before rotation: x = %d, y = %d, z = %d\n", x, y, z);
   // Assign addresses to pointers
   px = &x;
   py = &y;
   pz = &z;

   // Rotate values using pointers
   temp = *px;
   *px = *py;
   *py = *pz;
   *pz = temp;
   printf("After rotation: x = %d, y = %d, z = %d\n", x, y, z);
   return 0;
}
```

9. Print addresses and values of array elements using pointers

```
#include <stdio.h>
int main() {
   int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
   int *ptr;
   // Assign base address of array to pointer
```

```
    ptr = arr; // equivalent to ptr = &arr[0];
    printf("Array elements, addresses and values:\n");
    printf("Index\tAddress\t\tValue\n");
    for(int i = 0; i < 10; i++) {
        printf("%d\t%p\t%d\n", i, (ptr + i), *(ptr + i));
    }
    return 0;
}
```