Day : Pointers (9-8-2025)

1. 1. Print the address of a variable using pointer

```
#include <stdio.h>
int main() {
    int num = 25;
    int *ptr = &num;
    printf("Address of num variable: %p\n", ptr);
    return 0;
}
```

Output:
Address of num variable: 0x7ffee1a5f3cc

2. 2. Access array elements using pointers

```
#include <stdio.h>
int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int *ptr = arr;
    for (int i = 0; i < 5; i++) {
        printf("Element %d = %d\n", i, *(ptr + i));
    }
    return 0;
}
```

Output:
Element 0 = 10
Element 1 = 20
Element 2 = 30
Element 3 = 40
Element 4 = 50

3. 3. Swap two numbers using pointers

```
#include <stdio.h>
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
int main() {
    int x = 10, y = 20;
    printf("Before swap: x = %d, y = %d\n", x, y);
```

```
    swap(&x, &y);
    printf("After swap: x = %d, y = %d\n", x, y);
    return 0;
}
```

Output:
Before swap: x = 10, y = 20
After swap: x = 20, y = 10

4. 4. Add two numbers using pointers

```
#include <stdio.h>
void add(int *a, int *b, int *sum) {
    *sum = *a + *b;
}
int main() {
    int num1 = 15, num2 = 25, result;
    add(&num1, &num2, &result);
    printf("Sum = %d\n", result);
    return 0;
}
```

Output:
Sum = 40

5. 5. Find the length of a string using pointers

```
#include <stdio.h>
int stringLength(char *str) {
    int len = 0;
    while (*str != '\0') {
        len++;
        str++;
    }
    return len;
}
int main() {
    char str[] = "Hello, World!";
    int length = stringLength(str);
    printf("Length of string = %d\n", length);
    return 0;
}
```

Output:
Length of string = 13

6. 6. Reverse a string using pointers

```
#include <stdio.h>
#include <string.h>
void reverseString(char *str) {
    char *start = str;
    char *end = str + strlen(str) - 1;
    while (start < end) {
        char temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}
int main() {
    char str[] = "Pointer";
    reverseString(str);
    printf("Reversed string: %s\n", str);
    return 0;
}
```

Output:
Reversed string: retniop

7. 7. Count vowels using pointer

```
#include <stdio.h>
int countVowels(char *str) {
    int count = 0;
    while (*str != '\0') {
        char ch = *str;
        if (ch == 'a' || ch == 'e' || ch == 'i' ||
            ch == 'o' || ch == 'u' || ch == 'A' ||
            ch == 'E' || ch == 'I' || ch == 'O' ||
            ch == 'U') {
            count++;
        }
        str++;
```

```
    }
    return count;
}
int main() {
    char str[] = "Pointer Example";
    int vowels = countVowels(str);
    printf("Number of vowels: %d\n", vowels);
    return 0;
}
```

Output:
Number of vowels: 6

8. 8. Demonstrate pointer to pointer

```
#include <stdio.h>
int main() {
    int var = 3000;
    int *ptr = &var;
    int **pptr = &ptr;
    printf("Value of var = %d\n", var);
    printf("Value available at *ptr = %d\n", *ptr);
    printf("Value available at **pptr = %d\n", **pptr);
    return 0;
}
```

Output:
Value of var = 3000
*Value available at *ptr = 3000*
*Value available at **pptr = 3000*

9. 9. Allocate memory using malloc() and free it

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int *ptr;
    ptr = (int*) malloc(sizeof(int) * 5);
    if (ptr == NULL) {
        printf("Memory not allocated.\n");
        return 1;
    }
    for (int i = 0; i < 5; i++) {
```

```
        ptr[i] = i + 1;
    }
    printf("Allocated array elements: ");
    for (int i = 0; i < 5; i++) {
        printf("%d ", ptr[i]);
    }
    printf("\n");
    free(ptr);
    return 0;
}
```

Output:

Allocated array elements: 1 2 3 4 5

10. 10. Sort an array using pointer notation

```
#include <stdio.h>
void sortArray(int *arr, int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (*(arr+j) > *(arr+j+1)) {
                int temp = *(arr+j);
                *(arr+j) = *(arr+j+1);
                *(arr+j+1) = temp;
            }
        }
    }
}
int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    sortArray(arr, n);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", *(arr+i));
    }
    printf("\n");
    return 0;
}
```

Output:

Sorted array: 11 12 22 25 34 64 90