Day7 : Pointers (9-8-2025)

1.Write a program to print the address of a variable using pointer.

IPO:

INPUT: An integer value

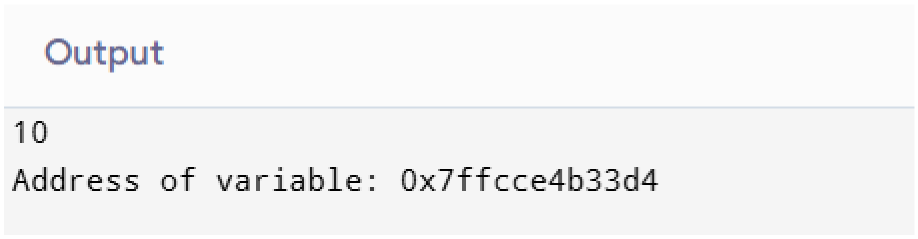
PROCESS: Store variable’s address in a pointer and display it

OUTPUT: Address of the variable

CODE;

```
#include <stdio.h>

void main()

{

    int num;

    int *ptr;

    scanf("%d", &num);

    ptr = &num;

    printf("Address of variable: %p", ptr);

}
```

OUTPUT;



```
10
Address of variable: 0x7ffcce4b33d4
```

2.Write a program to access array elements using pointers.

IPO:

INPUT: Elements of an array

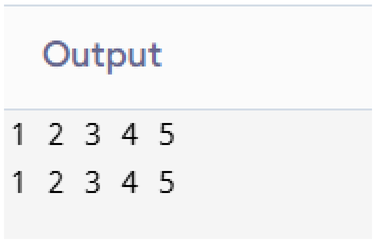
PROCESS: Use pointer arithmetic to access and print each element

OUTPUT: Array elements

CODE;

```
#include <stdio.h>
void main()
{
   int arr[5], i;
   int *ptr = arr;
   for(i = 0; i < 5; i++)
      scanf("%d", ptr + i);
   for(i = 0; i < 5; i++)
      printf("%d ", *(ptr + i));
}
```

OUTPUT;



```
1 2 3 4 5
1 2 3 4 5
```

3.Write a program to swap two numbers using pointers.

IPO:

INPUT: Two integers

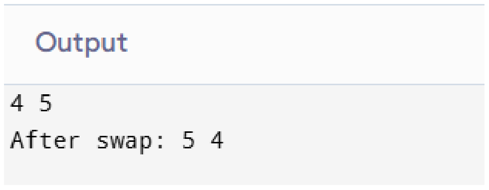
PROCESS: Swap using pointer dereferencing

OUTPUT: Swapped numbers

CODE;

```
#include <stdio.h>
void main()
{
    int a, b, temp;
    int *p = &a, *q = &b;
    scanf("%d%d", &a, &b);
    temp = *p;
    *p = *q;
    *q = temp;
    printf("After swap: %d %d", a, b);
}
```

OUTPUT;



```
4 5
After swap: 5 4
```

4.Write a program to add two numbers using pointers.

IPO:

INPUT: Two integers

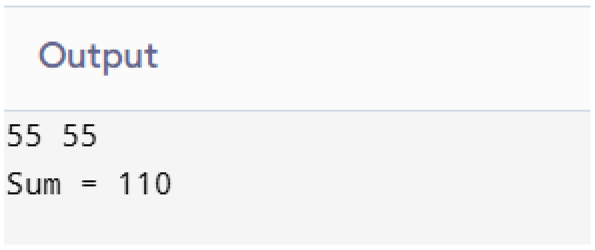
PROCESS: Use pointers to access and add values

OUTPUT: Sum of numbers

CODE;

```c
#include <stdio.h>

void main()

{

    int x, y, sum;

    int *p = &x, *q = &y;

    scanf("%d%d", &x, &y);

    sum = *p + *q;

    printf("Sum = %d", sum);

}
```

OUTPUT;



```
55 55
Sum = 110
```

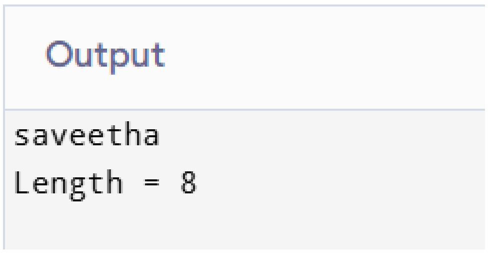
5.Write a program to find the length of a string using pointers.

INPUT: A string entered by the user (single word)
PROCESS: Start a pointer at the first character and count the string
OUTPUT: Display the total number of characters in the string

CODE;

```
#include <stdio.h>
void main()
{
    char str[100], *p;
    int len = 0;
    scanf("%s", str);
    p = str;
    while(*p != '\0')
    {
        len++;
        p++;
    }
    printf("Length = %d", len);
}
```

OUTPUT;



```
saveetha
Length = 8
```

6.Write a program to reverse a string using pointers.

IPO

INPUT: A string entered by the user

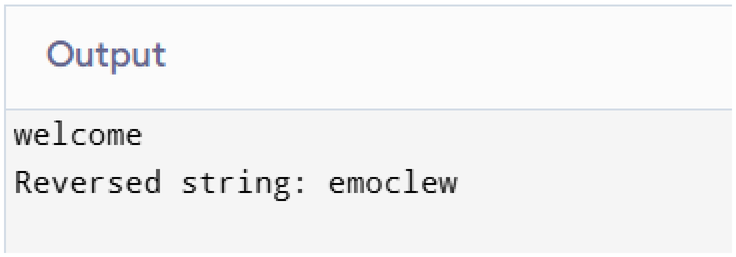
PROCESS: Use two pointers to swap first and last characters until middle is reached

OUTPUT: Reversed string

CODE;

```
#include <stdio.h>
void main()
{
   char str[100], *start, *end, temp;
   int len = 0;
   scanf("%s", str);
   while(str[len] != '\0')
      len++;
   start = str;
   end = str + len - 1;
   while(start < end)
   {
      temp = *start;
      *start = *end;
      *end = temp;
      start++;
      end--;
   }
   printf("Reversed string: %s", str);
}
```

OUTPUT;


Output

```
welcome
Reversed string: emoclew
```

7.Write a program to count vowels using pointer.

IPO:

INPUT: A string given by user

PROCESS: Use pointer to traverse and count vowels

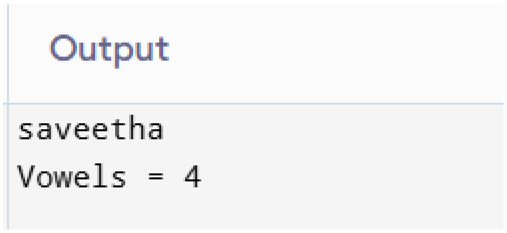
OUTPUT: Number of vowels

CODE;

```
#include <stdio.h>
void main()
{
   char str[100], *p;
   int vowels = 0;
   scanf("%s", str);
   p = str;
   while(*p != '\0')
   {
      if(*p=='a'||*p=='e'||*p=='i'||*p=='o'||*p=='u'||
         *p=='A'||*p=='E'||*p=='I'||*p=='O'||*p=='U')
          vowels++;
      p++;
   }
   printf("Vowels = %d", vowels);
```

}

OUTPUT;



```
saveetha
Vowels = 4
```

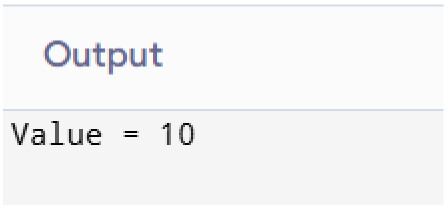
8.Write a program to demonstrate pointer to pointer.

IPO:
 INPUT: An integer
 PROCESS: Store address of variable in pointer, address of pointer in another pointer
 OUTPUT: Value using pointer to pointer

CODE;

```
#include <stdio.h>
int main()
{
   int num = 10;
   int *ptr = &num;
   int **pptr = &ptr;
   printf("Value = %d", **pptr);
}
```

OUTPUT;



```
Value = 10
```

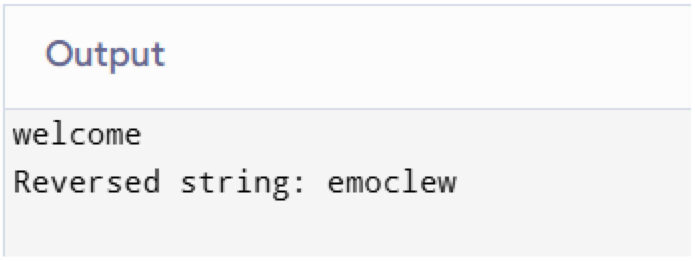
10.Write a program to sort an array using pointer notation.

IPO:
 INPUT: Elements of an array
 PROCESS: Use pointer arithmetic in bubble sort
 OUTPUT: Sorted array

CODE:

```
#include <stdio.h>
void main()
{
   char str[100], *start, *end, temp;
   int len = 0;
   scanf("%s", str);
   while(str[len] != '\0')
      len++;
   start = str;
   end = str + len - 1;
   while(start < end)
   {
      temp = *start;
      *start = *end;
      *end = temp;
      start++;
      end--;
   }
   printf("Reversed string: %s", str);
}
```

OUTPUT;



```
welcome
Reversed string: emoclew
```