DAY 9 ASSIGNMENT

Write a program in C that demonstrates the use of a pointer to a const array of integers. The program should do the following:

1. Define an integer array with fixed values (e.g., {1, 2, 3, 4, 5}).

```
#include<stdio.h>
int main()
{
    const int array[]={1,2,3,4,5};
    int *ptr = array;
    for(int i=0;i<5;i++){
        printf("%d",*(ptr+i));
    }

}
```

2

```
#include<stdio.h>
int main()
{
   int const array[]={1,2,3,4,5};
    int *ptr = array;
    // *ptr+3=2;
    for(int i=0;i<5;i++){
        printf("%d",*(ptr+i));
    }

}
```

3 Implement a function printArray(const int *arr, int size) to print the elements of the array using the const pointer.

```
#include<stdio.h>
int values( const int * arr,int size);
int main(){
   const int array[] = {1,2,3,4,5,6,7,8};
   values(array,8);
}
int values(const int * arr,int size){
    for(int i=0;i<size;i++){
```

```
        printf("%d\n",*(arr+i));
    }
}
```

4. Attempt to modify an element of the array through the pointer (this should produce a compilation error, demonstrating the behavior of const).

```
#include<stdio.h>
int values( const int * arr,int size);
int main(){
   const int array[] = {1,2,3,4,5,6,7,8};
   values(array,8);

   const int *ptr =array
   *(ptr+1)=6;// we cannot modify this
}
int values(const int * arr,int size){
    for(int i=0;i<size;i++){
        printf("%d\n",*(arr+i));
    }
}
```

Write a program in C that demonstrates the use of a pointer to a const integer and a const pointer to an integer. The program should:

1. Define an integer variable and initialize it with a value (e.g., int value = 10;).

```
#include<stdio.h>

int main() {
    int value = 10;
    const int *ptrToConst = &value;

    printf("Pointer to a const integer:\n");
    printf("Value accessed through ptrToConst: %d\n", *ptrToConst);
    value = 20;
    printf("Modified value directly: %d\n", *ptrToConst);

    int *const constPtr = &value;

    printf("\nConst pointer to an integer:\n");
    printf("Value accessed through constPtr: %d\n", *constPtr);
```

```
    *constPtr = 40;
    printf("Modified value through constPtr: %d\n", value);

    return 0;
}
```

6. Create a pointer to a const integer and demonstrate that the value cannot be modified through the pointer.

```
#include<stdio.h>
int main(){
    int value =10;
    const int *ptr =&value;
    printf("%d\n",*ptr);
    // *ptr =20;
    value=30;
    printf("%d\n",*ptr);
}
```

7

```
#include<stdio.h>
int main(){
    int value =10;
    int *const ptr =&value;
    printf("%d\n",*ptr);

    *ptr =30;
    printf("%d\n",value);
}
```

8. Print the value of the integer and the pointer address in each case.

```
#include<stdio.h>
int main(){
    int value =10;
    int *const ptr = &value;
    printf("%p\n",value);
    printf("%p\n",*ptr);
    *ptr =20;
     printf("%p\n",value);
```

```
    printf("%p\n",*ptr);
}
```

9

a. Use the type qualifiers const int* and int* const appropriately.

b. Attempt to modify the value or the pointer in an invalid way to show how the compiler enforces the constraints.

```
#include<stdio.h>
int main(){
    int value1=10;
    int value2 = 20;
    const int *ptr = &value1;
    int*const ptr1 =&value2;
    printf("the address %d\n",*ptr);
    printf("the address %d\n",*ptr1);
    printf("%p\n",ptr);
    // *ptr=20;
    *ptr1=30;
    // ptr1=&value1;
    ptr=&value2;
    printf("%d\n",*ptr1);
    printf("%p\n",ptr);
}
```

Problem: Universal Data Printer

You are tasked with creating a universal data printing function in C that can handle different types of data (int, float, and char*). The function should use void pointers to accept any type of data and print it appropriately based on a provided type specifier.

Specifications

Implement a function print_data with the following signature:

```
void print_data(void* data, char type);
```

Parameters:

data: A void* pointer that points to the data to be printed.

type: A character indicating the type of data:

- 'i' for int
- 'f' for float
- 's' for char* (string)

Behavior:

If type is 'i', interpret data as a pointer to int and print the integer.
If type is 'f', interpret data as a pointer to float and print the floating-point value.
If type is 's', interpret data as a pointer to a char* and print the string.

In the main function:

Declare variables of types int, float, and char*.
Call print_data with these variables using the appropriate type specifier.

Example output:
Input data: 42 (int), 3.14 (float), "Hello, world!" (string)
Output:
Integer: 42
Float: 3.14
String: Hello, world!

Constraints
1. Use void* to handle the input data.
2. Ensure that typecasting from void* to the correct type is performed within the print_data function.
3. Print an error message if an unsupported type specifier is passed (e.g., 'x').

```
#include<stdio.h>
void print_val(void*data,char type);
int main(){
   int ival=42;
   float fval=3.14;
   char *string="hello world!";
   print_val(&ival,'i');
   printf("\n");
   print_val(&fval,'f');
   printf("\n");
   print_val(&string,'s');
   printf("\n");

return 0;
}
void print_val (void*data,char type){

   if(type=='i'){
      printf("%d",*(int*)data);
   }
```

```
    else if(type=='f'){
        printf("%.2f",*(float*)data);
    }
    else if(type=='s'){
        printf("%s",*(char**)data);
    }
}

----------------------------------------
#include <stdio.h>

void add(char result[], const char str[], const char str2[]) ;
int my_strcmp( char *str, char *str2);

int main() {
    char result[100];
    const char str[] = "Hello, ";
    const char str2[] = "World";
    add(result, str, str2);

    printf("Concatenated String: %s\n", result);
    int res = my_strcmp(str,str2);
    if(res == 0)
    {
        printf("Not equal\n");
    }
    else
    {
        printf("Equal\n");
    }
    return 0;
    return 0;
}
void add(char result[], const char str[], const char str2[]) {
    int i = 0;
    while (str[i] != '\0') {
        result[i] = str[i];
        i++;
    }
    int j = 0;
    while (str2[j] != '\0') {
        result[i] = str2[j];
        i++;
        j++;
```

```
    }
}
int my_strcmp( char *str1, char *str2)
{

    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0')
    {
        if (str1[i] != str2[i]) {
            return 0;
        }
        i++;
    }
    return 1;
}
```

CLASS WORKS

```
#include<stdio.h>
int main(){
    int a[] = {1,2,3,4,5,6,7,8,9};
    int *ptr =a;
    for(int i=0;i<9;i++){
        printf("a[%d]= %d\n",i,*(ptr+i));
    }
    printf("\n");
    *(ptr +6)=8; //this  modifing the array and store value in arr 8

    for(int i=0;i<9;i++){
        printf("a[%d]= %d\n",i,*(ptr+i));
    }

}
```

```
#include<stdio.h>
int main(){
    int a[]={1,2,3};
    int *ptr =a; //*ptr=&a we can write this way also
    printf("address of the array a[]=%p\n",a);
    printf("ptr =%p\n",ptr);
```

```
}
-------------------------------------------------


#include<stdio.h>
int main(){
    int a[]={1,2,3};
    printf("address of the array a[]=%p\n",a);
    printf("the element of the 0th index =%d\n",a[0]);
    printf("the element of the 0th index =%d\n",*(a+0));
    printf("address of the array a[]=%p\n",a+1);
    printf("the element of the 0th index =%d\n",a[1]);
    printf("the element of the 0th index =%d\n",*(a+1));
    int *ptr = &a[0];
    printf("address of the array a[]=%p\n",a);
    printf("ptr =%p\n",ptr);

}
-----------------------------------------------------------
#include<stdio.h>

int addArray(int *array,int n);
int main(){
    int a[10] = {0,1,2,3,4,5,6,7,8,9};
    int sum =0;
    sum = addArray(a,10); //&a[0]
    printf("the sum is %d",sum);
    return 0;
}
int addArray(int *array,int n){
    int arsum=0;
    for(int i=0;i<n;i++){
        arsum = arsum + *(array + i);

    }
    return arsum;
}
-----------------------------------------------------------------
#include<stdio.h>
int main(){
    int a[]={1,2,3};
    printf("address of the array a[]=%p\n",a);
    printf("Address of a[1]=%d\n",a+1);
    printf("Address of a[2]=%d\n",a+2);
```

```
}
------------------------------------------------------------
STRING

#include<stdio.h>
int main(){
    char name[6] ={"anusree"};
    for(int i=0;i<6;i++){
        printf("%c\n",name[i]);
    }
    printf("size of name =%d",sizeof(name));
}
------------------------------------------------------------------
#include<stdio.h>
int main(){
    char name[50];
    printf("enter the name");
    scanf("%s",name);
    printf("%s",name);
}
------------------------------------------------------------------
#include<stdio.h>
int main(){
    char str1[]="hi anusree";
    char str2[]="hi arya";
    int count =0;
    while(str1[count]!='\0'){
        count++;
    }

    printf("the length is %d\n",count);
    count =0;
     while(str2[count]!='\0'){
        count++;
    }
    printf("the length is %d\n",count);
}
-----------------------------------------------
#include<stdio.h>
int main(){
    int i=10;
    float f=2.34;
    char ch ='k';
```

```
    void *ptr;
    ptr = &i;
    printf("i=%d\n",*(int *)ptr);
    ptr = &f;
    printf("f=%f\n",*(float *)ptr);
    ptr =&ch;
    printf("c=%c\n",*(char *)ptr);
}
```