----------Pointers----------

```
//1
#include<stdio.h>
int main(){
   int a=10;
   int *pint=&a;
   printf("the value of a = %d",*pint);
}

//2
#include<stdio.h>
int main(){
   float a=20;
   float *pflt=&a;
   *pflt+=10;
   float n=*pflt;
   printf("the changed value is = %f",n);
}

//3
#include<stdio.h>
int sum(int*,int);
int sum(int arr[],int size){
   int total=0;
   for(int* ptr=arr;ptr<arr+size;ptr++){
      total+=*ptr;
   }
   return total;
}
```

```
int main(){
   int arr[]={1,2,3,4,5};
   int size=sizeof(arr)/sizeof(arr[0]);
   int result=sum(arr,size);
   printf("the sum is %d",result);

}

//4
#include<stdio.h>

int main() {
   int *pint = NULL;  // Declare a pointer and assign it a NULL value

   // Check if the pointer is NULL before dereferencing it
   if (pint == NULL) {
      printf("The pointer is NULL, cannot dereference it.\n");
   } else {
      // This part won't run since pint is NULL
      printf("The value pointed to by pint is: %d\n", *pint);
   }

   return 0;
}

//5
#include<stdio.h>

int main() {
```

```
    int *pint;  // Declare an uninitialized pointer

    // Attempt to dereference the wild pointer
    printf("The value pointed to by pint is: %d\n", *pint);  // Undefined behavior

    return 0;
}

//6
#include<stdio.h>
int main(){
    int a=10;
    int *ptr1=&a;;
    int **ptr2=&ptr1;
    printf("Using p2 (pointer to pointer): %d\n", **ptr2);
    printf("Using p1 (pointer to int): %d\n", *ptr1);
    printf("Directly using variable 'a': %d\n", a);
}

//7
#include<stdio.h>
#include<stdlib.h>
int main(){
    int size=5;

    int *arr=(int*)malloc(size*sizeof(int));
    if(arr==NULL){
        printf("Memory allocation falied!");
        return 1;
```

```
    }
    for(int i=0;i<size;i++){
        printf("enter the value for element %d",i+1);
        scanf("%d",&arr[i]);
    }
    for(int i=0;i<size;i++){
        printf("The element %d id %d\n",i+1,*(arr+i));
    }
    free(arr);
    return 0;

}

//8

#include<stdio.h>
int sum(int a,int b){
    return a+b;
}

int main(){
    int a,b;
    printf("enter the value for a and b");
    scanf("%d%d",&a,&b);
    int(*func_ptr)(int,int)=sum;
    int result=func_ptr(a,b);
    printf("The sum of %d and %d is: %d\n", a, b, result);
    return 0;
}
```

```
//9

#include <stdio.h>

int main() {
   int a = 5;
   int b = 10;
   int *const ptr = &a;
   printf("Value pointed to by ptr: %d\n", *ptr);
   *ptr = 20;
   printf("After modifying the value, ptr points to: %d\n", *ptr);
   return 0;
}
#include <stdio.h>

int main() {
   int a = 5;
   int b = 10;
   const int *ptr = &a;
   printf("Value pointed to by ptr: %d\n", *ptr);
   ptr = &b;
   printf("After changing ptr to point to b, ptr points to: %d\n", *ptr);  // Output: 10
   return 0;
}

//10
#include<stdio.h>
int main(){
```

```
    int a=10,b=20;
    int *p1=&a;
    int *p2=&b;
    if(p1<p2){
        printf("Pointer p1 (pointing to a) points to a lower memory address than p2 (pointing to b).\n");
    }else if(p1>p2){
        printf("Pointer p1 (pointing to a) points to a higher memory address than p2 (pointing to b).\n");
    }else{
        printf("Both pointers p1 and p2 point to the same memory address.\n");
    }
    return 0;
}
```

1.

```
#include<stdio.h>
int main(){
    int a=10;
    int b=20;
    int *const ptr=&a;
    printf("the pointer currently points to the adress of %p\n",ptr);
    *ptr=30;
    printf("the pointer now holds the data %d",*ptr);
    ptr=&b;
    printf("the changed adress of ptr is %p",ptr);//returns error
}
```

2.

```
#include<stdio.h>
```

```
int main(){
   int a=10;
   int const *ptr=&a;
   printf("The data that the pointer points is %d",*ptr);
   *ptr=40;//returns error as it cannot be modified
   printf("the data that the pointer points is %d",*ptr);
}
```

3.

```
#include<stdio.h>
int main(){
   int a=10;
   int b=20;
   int const *const ptr =&a;
   printf("the current adress that the pointer is holding is : %p",ptr);
   printf("the current data the pointer is pointing is %d",*ptr);
   //trying to change the value which will return the error
   *ptr=30;
   printf("the current data which pointer is pointing is %d",*ptr);
   ptr=&b;
  /printf("the current adress that pointer points is %p",ptr);
}
```

4.

```
#include<stdio.h>
int main(){
   int a=10,b=20,c=30;
   int *const ptr=&a;
   printf("Before modification:\n");
```

```
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    *ptr=100;
    printf("After modification :\n");
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    b=200;
    c=300;
     printf("After modifying 'b' and 'c' directly:\n");
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    return 0;

}
```

5.

```
#include<stdio.h>
int main(){
    int a,b;
    printf("enter the value for a :");
    scanf("%d",&a);
    printf("enter the value for b :");
    scanf("%d",&b);
    int *p1=&a;
    int *p2=&b;
    if(*p1==*p2){
        printf("the numbers are equal");
    }else if(*p1>*p2){
        printf("number 1 is bigger");
    }else{
        printf("number 2 is bigger");
    }
```

```
    return 0;
}
```

6.

```
#include<stdio.h>
int main(){
    int a=10,n;
    int *ptr=&a;
    printf("enter a value to point :");
    scanf("%d",&n);
    if(*ptr==n){
        printf("The pointer points to the value given by you!");
    }else{
        printf("The point is not pointing to the value given by you");
    }
    return 0;
}
```

7.

```
#include<stdio.h>
int main(){
    int a=10,b=20;
    int *pt1=&a;
    int *pt2=&b;
    if(pt1>pt2){
        printf("a's adress is larger than b's");
    }else{
        printf("b's adress is larger than a's");
    }
```

```
    return 0;
}
```

8.

```
#include<stdio.h>
int main(){
    int i=0;
    int *const pt1=&i;
    for(i=0;i<5;i++){
        *pt1=i;
        printf("the pointer now holds the data %d\n",*pt1);
    }
    return 0;
}
```

9.

```
#include <stdio.h>

int main() {
    int a = 10, b = 20, c = 30;
    int *const ptr = &a;
    int *ptr_arith = ptr;
    printf("Before modification:\n");
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    *ptr_arith = 100;
    printf("After modifying 'a' through dereferencing ptr:\n");
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    ptr_arith++;
    *ptr_arith = 200;
```

```
    printf("After modifying 'b' through dereferencing ptr:\n");
    printf("a = %d, b = %d, c = %d\n", a, b, c);
    ptr_arith++;
    *ptr_arith = 300;
    printf("After modifying 'c' through dereferencing ptr:\n");
    printf("a = %d, b = %d, c = %d\n", a, b, c);

    return 0;
}
```