```
// Author - Harsh Dixit
// Mail - harsh02.dixit@gmail.com
// Linkedin - https://www.linkedin.com/in/harsh-dixit10


// Pointers are the variables that store the address of another variable.
// Pointers only store the address. Any value other than address can't be stored.

#include<iostream>
using namespace std;
int main()
{
	int a = 5;
	int *ptr = &a;  // Here int * shows pointer to int data | &a shows adress of a | ptr is pointer name
	cout << a << endl; // gives value of a
	cout << ptr << endl; // gives address of a
	cout << *ptr << endl; // gives value at address of a
	cout << &ptr << endl; // gives address of ptr
	cout << &a << endl;

	/* char *ch ---> ch is pointer to character data
	bool *flag ---> flag is pointer to boolean data*/



	/* Access ---> accessing value stored at adress stored in pointer
	that value can be accessed using de-refrence operator. Its symbol is '*'.



	DIFFERENCE BETWEEN REFRENCE VARIABLE & POINTER
	---> Refrence variable is the diff name given to same memory location.It doesn't take extra space.
	---> Pointer stores address of another variable and it takes additinal space. */



	/* SIZE OF POINTER
	Size of pointer depends on variety of factors:
	1) Target platform's architechture
	2) Compiler
	3)Implementation and memory organization
	*/

	int  x = 10;
	int *ptr = &x;
	cout << "Size: " << sizeof(ptr) << endl;
	char ch = 'a';
	char *cptr = &ch;
	cout << "Size: " << sizeof(cptr) << endl;
	bool flag = 1;
	bool *bptr = &flag;
	cout << "Size: " << sizeof(bptr) << endl;




	//  POINTER DECLARATION
```

```
//      int *ptr;   //BAD PRACTICE
//      cout << *ptr << endl;
        /* Declaring pointer like this is bad practice as it is trying
        to access a random memory which is not allowed.
        Generally it gives run time error or segmentation fault.
        */

//      int *ptr = 0;
            // or
//      int *ptr = nullptr;

        /* create a null pointer to avoid above case.
        This also gives run time and segentation error but this will help
        while debugging the code.*/

// QUESTION 1

        int a = 100;
        int *ptr = &a;
        a = a + 1;
        ptr = ptr + 1;
        cout << a << " " << ptr;



// QUESTION 2

    int b = 10;
    int *ptr = &b;
    b = b + 1;
    cout << b << endl;
    *ptr = *ptr + 1;
    cout << *ptr ;



// QUESTION 3

    int a = 100;
    int *ptr  =  &a;
    cout << a << endl;  // value of a
    cout << &a << endl;  // address of a
    cout << ptr << endl; // address of a
    cout << *ptr << endl; // value at address of a or value of a
    cout << &ptr << endl; // address of pointer variable
    cout << *a << endl; // error (integer cant be de-refrence)
    (*ptr)++;
    cout << (*ptr) << endl; // value of a + 1
    ++(*ptr);
    cout << (*ptr) << endl; // value of a +1
    (*ptr) = (*ptr)/2;
    cout << (*ptr) << endl; // value of a/2
    *ptr = *ptr - 2;
    cout << *ptr << endl; // value of a -2



    // COPYING POINTER --->
```

```
    int a = 5;
    int *p = &a;
    int *q = p;
    cout << a << endl;
    cout << &a << endl;
    cout << *p << endl;
    cout << *q << endl;
    cout << &p << endl;
    cout << &q << endl;

    // *p and *q both are pointing to the value of a


    // POINTER WITH ARRAY
    // arr , &arr , &arr[0]  ---> return base address

    /* In case of int arrays
    int *ptr = arr;  ---> CORRECT
    int *ptr = &arr[0]; ---> CORRECT
    int *ptr = &arr; ---> INCORRECT
       */



// QUESTION 1

    int arr[5] = {10,20,30,40,50};
    int *ptr = arr;

    cout << arr << endl;
    cout << &arr[0] << endl;
    cout << ptr << endl;
    cout << &arr << endl;
    cout << *arr << endl;
    cout << *arr + 1 << endl;
    cout << *(arr) + 1 << endl;
    cout << *(arr + 2) << endl;
    cout << *(arr + 3) << endl;


    /* NOTE:-
    *(arr + 0) = arr[0]
    *(arr + 1) = arr[1]
    *(arr + 2) = arr[2]

    In general *(arr + i) = arr[i]
    */



// QUESTION 2


    int arr[5] = {10,20,30,40,50};
    int *p = arr;
    int *q = arr + 1;
    cout << p << endl;
    cout << &p << endl;
    cout << *p << endl;
    cout << q << endl;
```

```
    cout << &q << endl;
    cout << *q << endl;
    cout << *p + 1 << endl;
    cout << *(p) + 2 << endl;
    cout << *(q) + 2 << endl;
    cout << *(q) + 4 << endl;
    cout << *(q + 4) << endl; // garbage value
    cout << sizeof(arr) << endl;
    cout << sizeof(p) << endl;



    /* POINTER WITH CHAR ARRAY
    in case of char array -->
       char *cptr = ch; CORRECT
       char *cptr = &ch[0];  CORRECT
       char *cptr = &ch;  INCORRECT
       */



// Question 1

    char ch[50] = "Harsh";
    char *cptr = ch;
    cout << ch << endl;
    cout << &ch << endl;
    cout << *ch << endl;
    cout << ch[0] << endl;
    cout << &cptr << endl;
    cout << *cptr << endl;  // *cptr = *(cptr + 0 ) --> cptr[0]  --> H
    cout <<cptr << endl;

    /* NOTE ---->
    When we make a pointer for char array data and tries to print that pointer
    it will print the array character rather than its address like we saw in int array.*/

    // for example

    int arr[4] = {1,2,3,4};
    int *ptr = arr;
    cout << ptr << endl; // this gives the base address of arr
    char ch[10] = "Harsh";
    char *cptr = ch;
    cout << cptr << endl; // this will print all char in the char array




//Question 2

    char ch[30] = "ENORMOUS";
    char *cptr = ch;
    cout << ch << endl;
    cout << &ch << endl;
    cout << *(ch + 3) << endl;
    cout << cptr << endl;
    cout << &cptr << endl;
    cout << *(cptr + 3) << endl;
    cout << cptr + 2 << endl;
```

```
    cout << *cptr << endl;
    cout << cptr + 7 << endl;




// QUESTION 3

    char ch = 'M';
    char *cptr = &ch;
    cout << cptr << endl; // Some random char are printing here after M till we get null
char
    cout << *cptr << endl;

       return 0;
}
```