# Home Work # 2

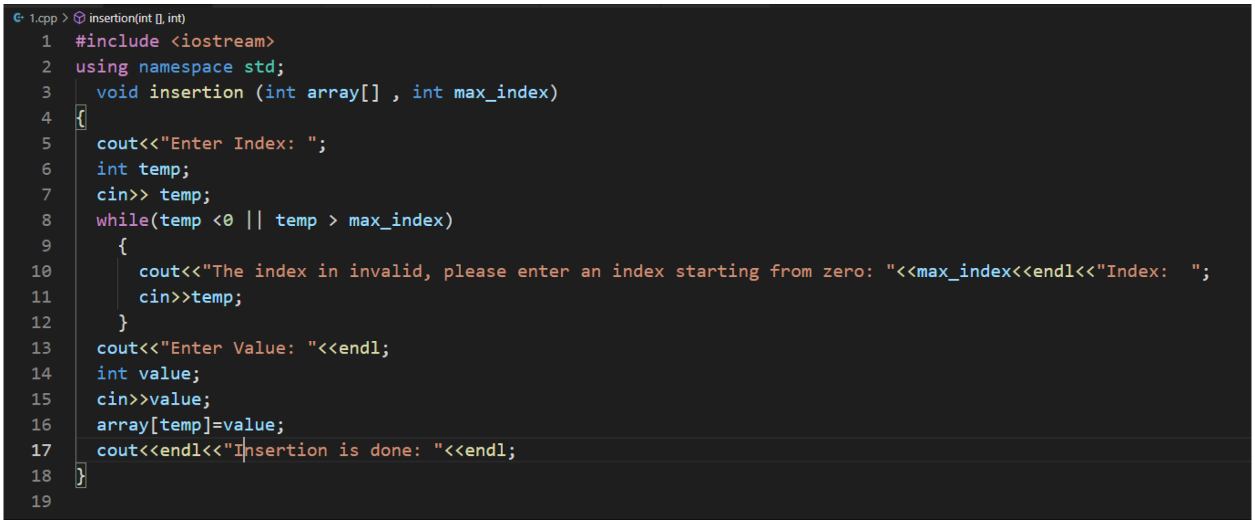
**Name:** Muhammad Sherjeel Akhtar
**Roll No:** 20P-0101

**Section:** BCS-4B

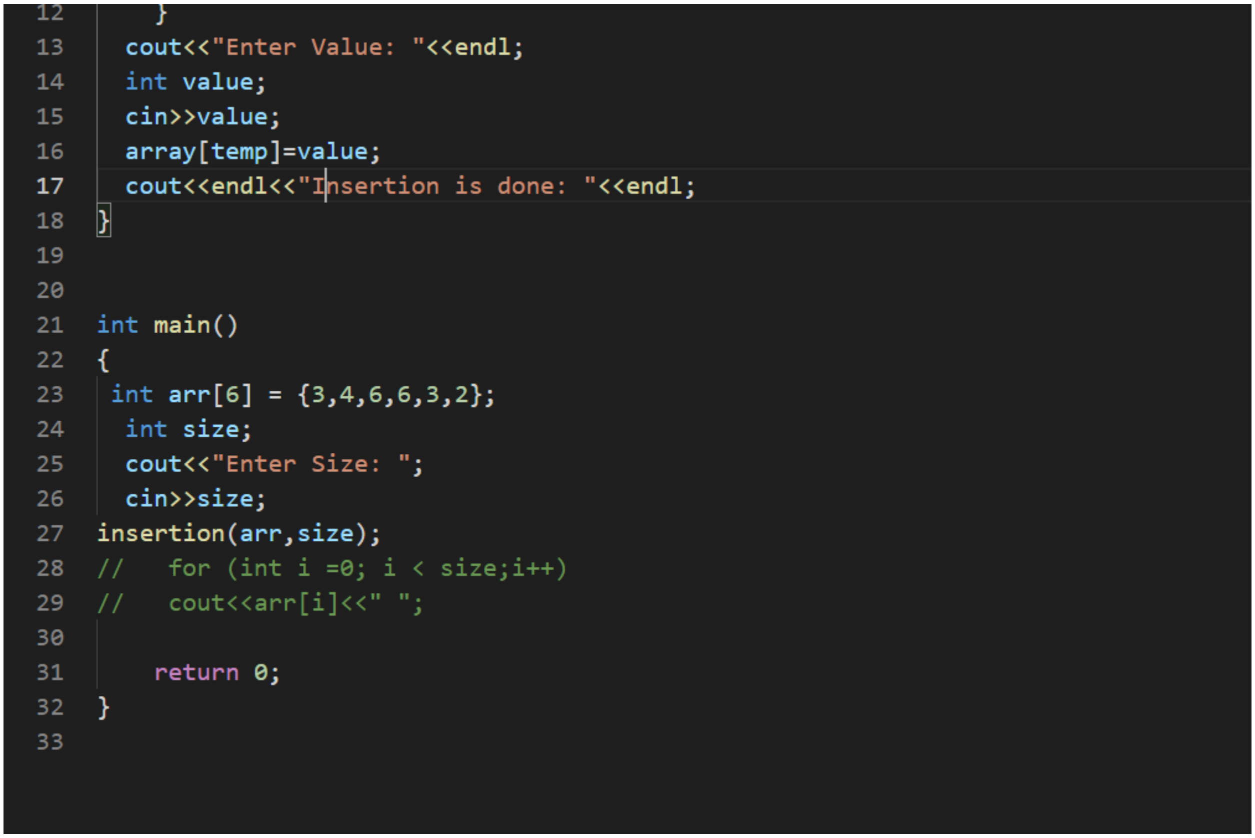
**Homework No:** 2

**Submitted To Respected Sir:** Khurram Shehzad

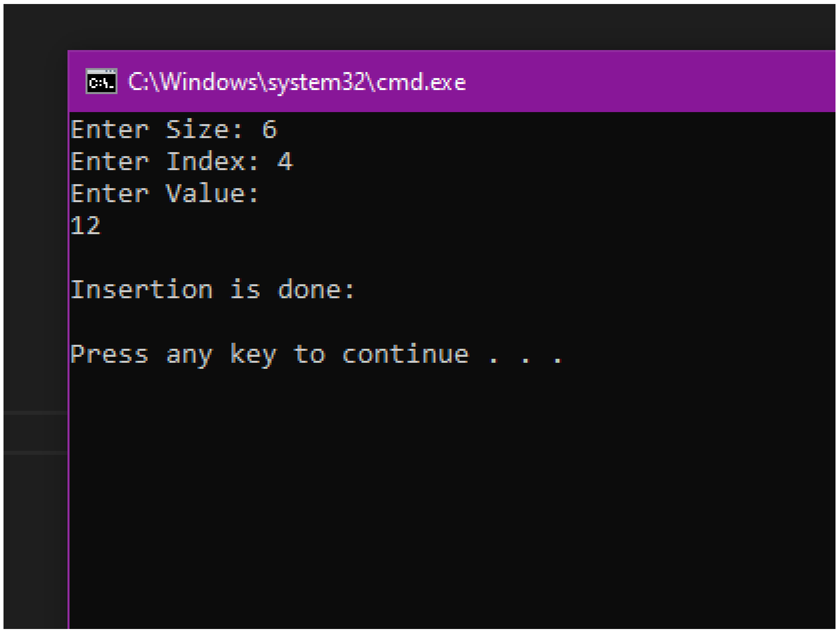
# Question 1:



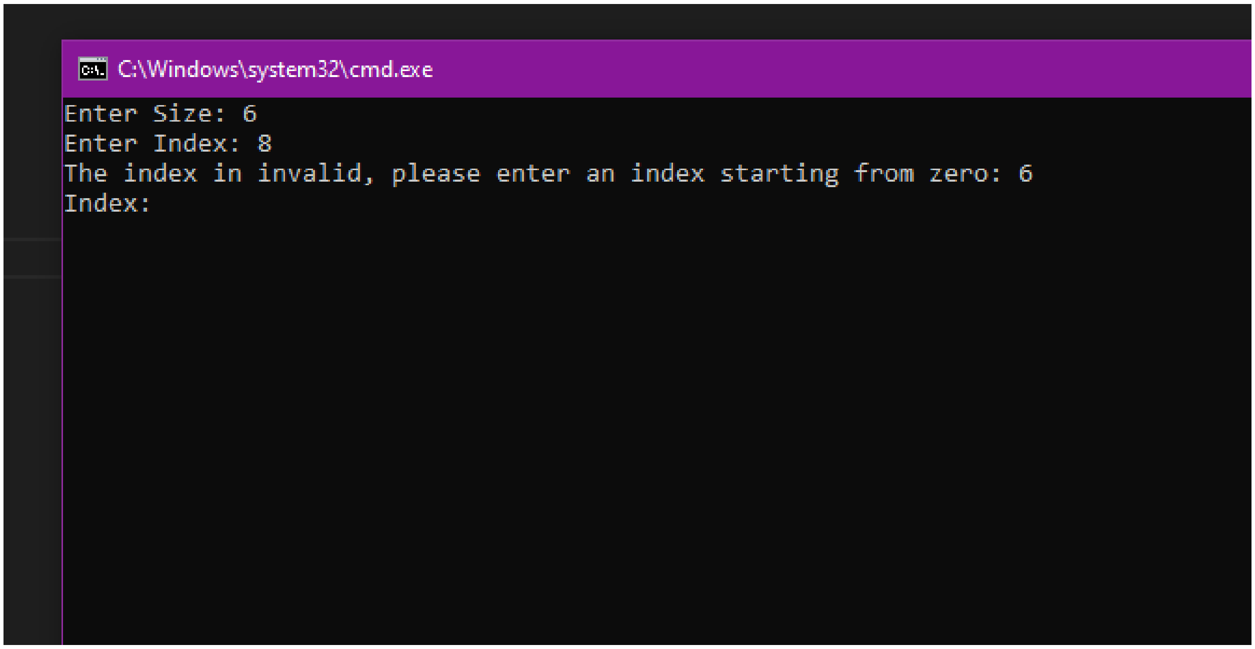
```cpp
#include <iostream>
using namespace std;
  void insertion (int array[] , int max_index)
{
  cout<<"Enter Index: ";
  int temp;
  cin>> temp;
  while(temp <0 || temp > max_index)
    {
      cout<<"The index in invalid, please enter an index starting from zero: "<<max_index<<endl<<"Index:  ";
      cin>>temp;
    }
  cout<<"Enter Value: "<<endl;
  int value;
  cin>>value;
  array[temp]=value;
  cout<<endl<<"Insertion is done: "<<endl;
}
```



```
    }
  cout<<"Enter Value: "<<endl;
  int value;
  cin>>value;
  array[temp]=value;
  cout<<endl<<"Insertion is done: "<<endl;
}


int main()
{
 int arr[6] = {3,4,6,6,3,2};
  int size;
  cout<<"Enter Size: ";
  cin>>size;
insertion(arr,size);
//   for (int i =0; i < size;i++)
//   cout<<arr[i]<<" ";

    return 0;
}
```



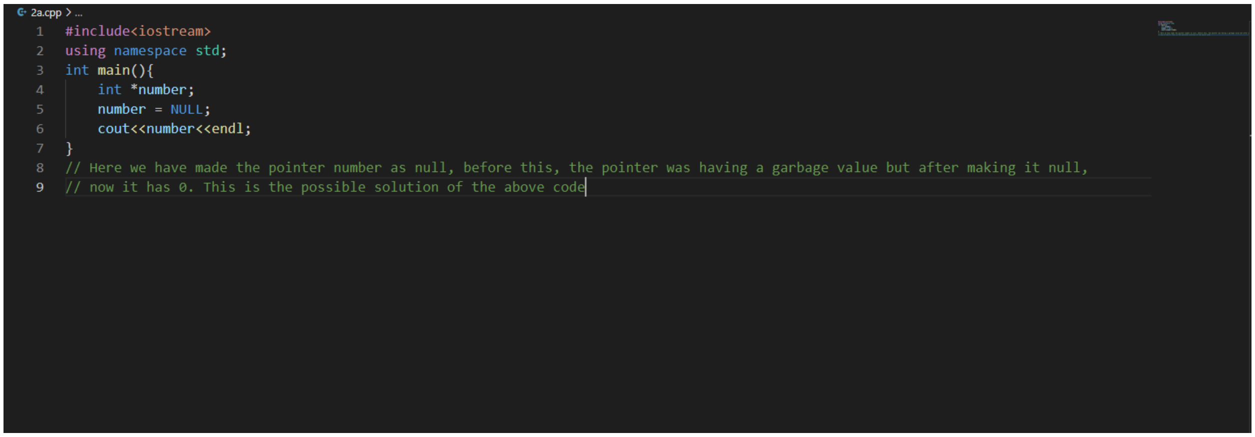
```
Enter Size: 6
Enter Index: 4
Enter Value:
12

Insertion is done:

Press any key to continue . . .
```



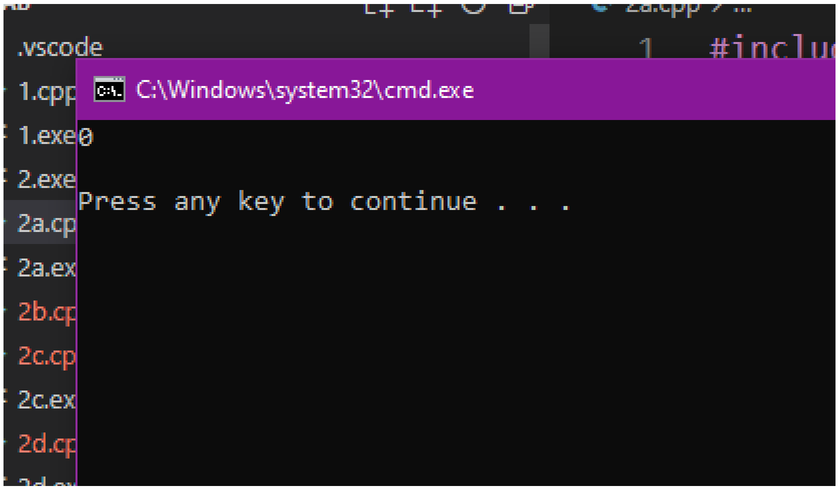
```
Enter Size: 6
Enter Index: 8
The index in invalid, please enter an index starting from zero: 6
Index:
```

# Question 2:

## A:

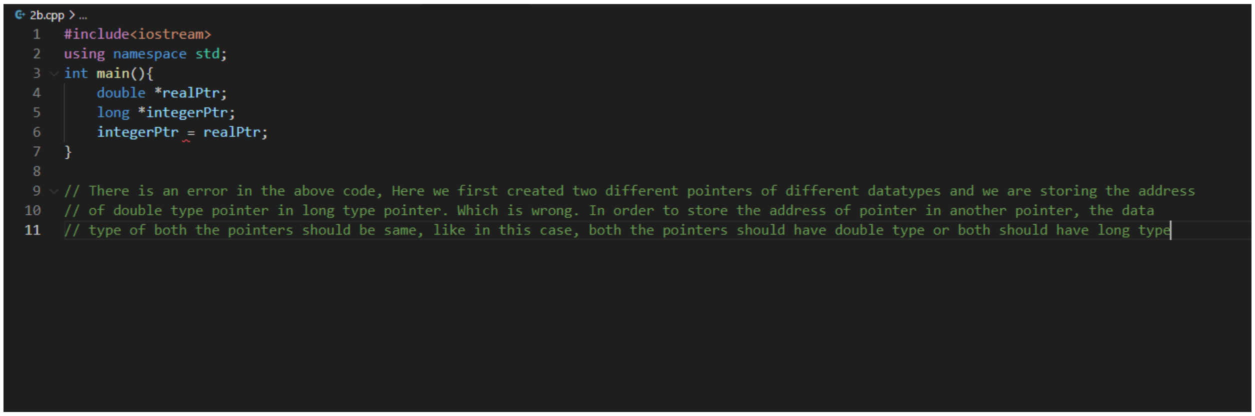


```cpp
#include<iostream>
using namespace std;
int main(){
    int *number;
    number = NULL;
    cout<<number<<endl;
}
// Here we have made the pointer number as null, before this, the pointer was having a garbage value but after making it null,
// now it has 0. This is the possible solution of the above code
```

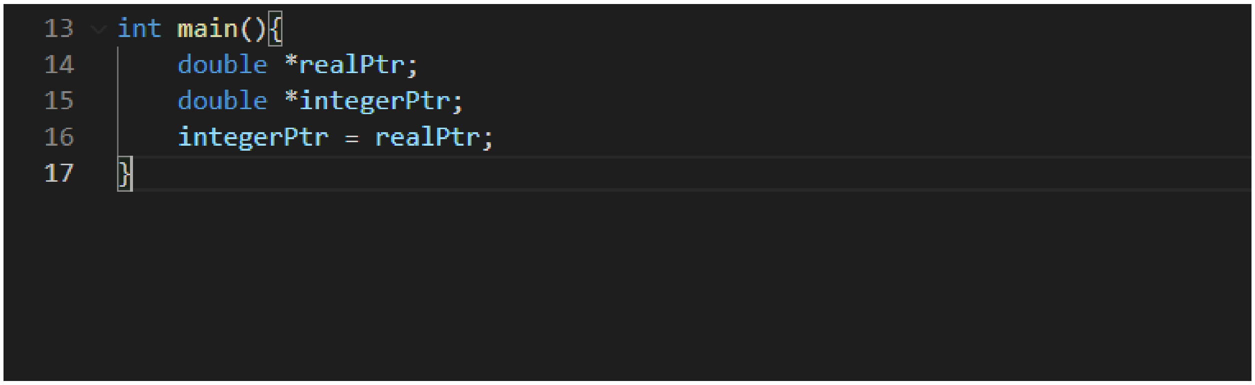


```
0
Press any key to continue . . .
```

## B:

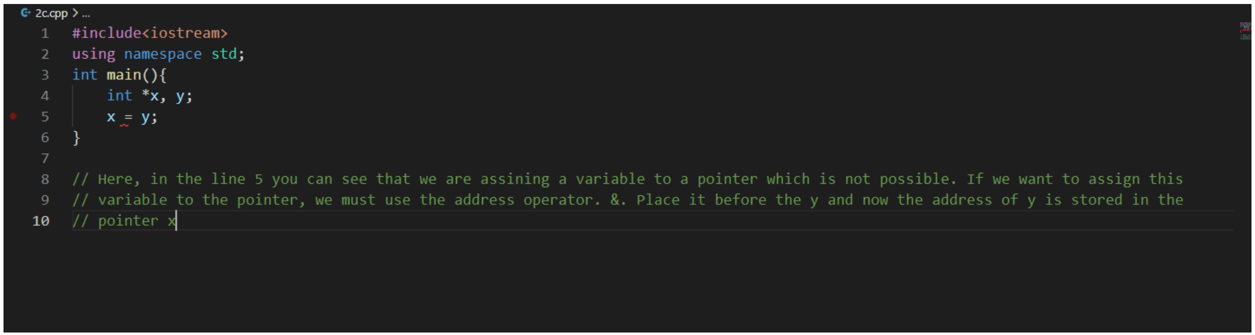


```cpp
#include<iostream>
using namespace std;
int main(){
    double *realPtr;
    long *integerPtr;
    integerPtr = realPtr;
}

// There is an error in the above code, Here we first created two different pointers of different datatypes and we are storing the address
// of double type pointer in long type pointer. Which is wrong. In order to store the address of pointer in another pointer, the data
// type of both the pointers should be same, like in this case, both the pointers should have double type or both should have long type
```

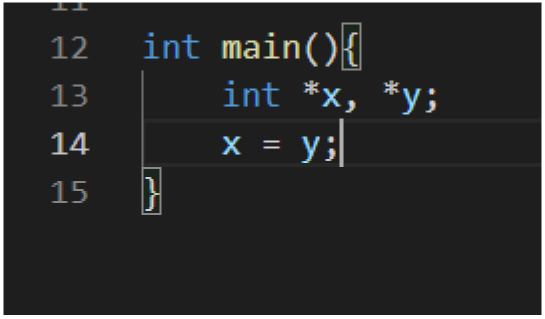


```cpp
int main(){
    double *realPtr;
    double *integerPtr;
    integerPtr = realPtr;
}
```

## C:

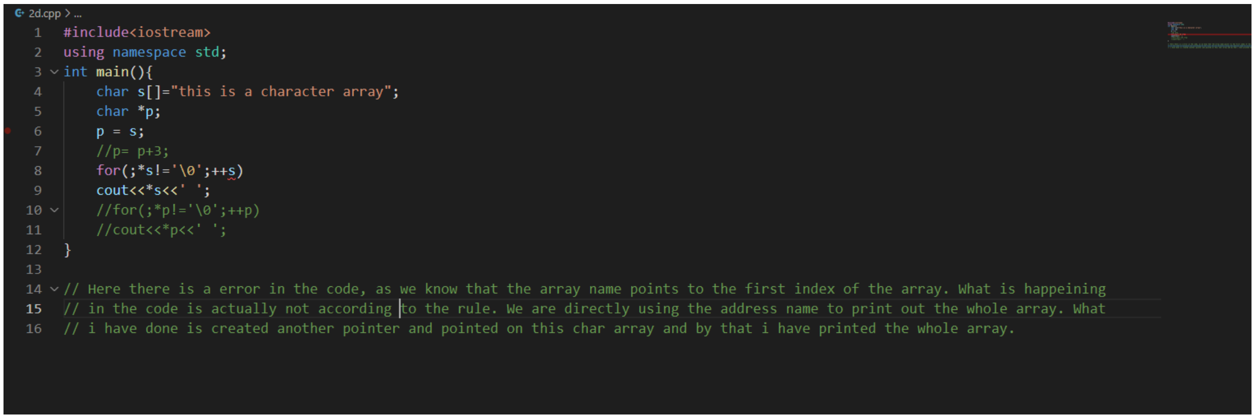


```
1   #include<iostream>
2   using namespace std;
3   int main(){
4       int *x, y;
5       x = y;
6   }
7
8   // Here, in the line 5 you can see that we are assining a variable to a pointer which is not possible. If we want to assign this
9   // variable to the pointer, we must use the address operator. &. Place it before the y and now the address of y is stored in the
10  // pointer x
```

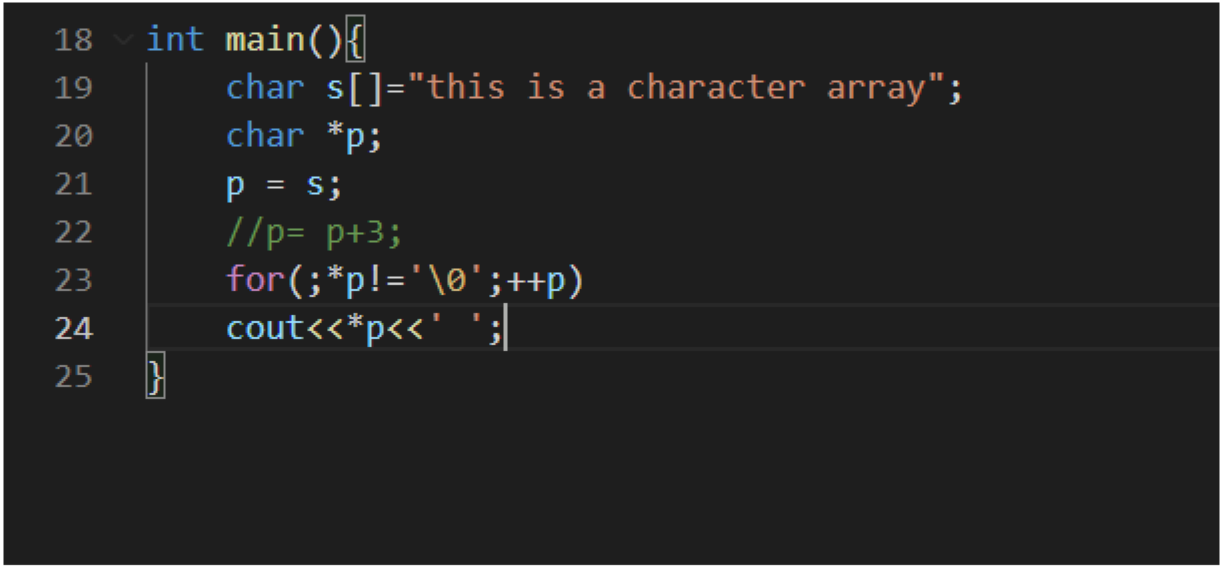


```
int main(){
    int *x, *y;
    x = y;
}
```

# D:

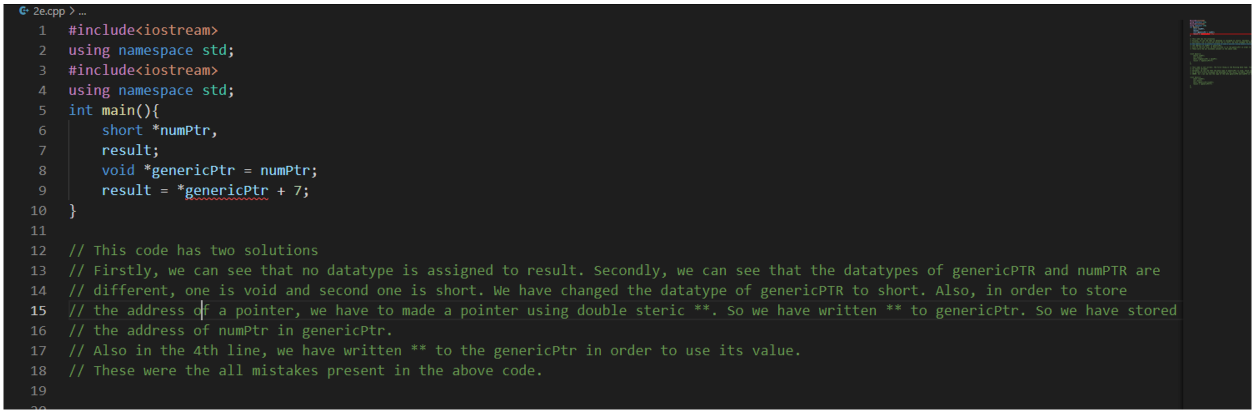


```cpp
#include<iostream>
using namespace std;
int main(){
    char s[]="this is a character array";
    char *p;
    p = s;
    //p= p+3;
    for(;*s!='\0';++s)
    cout<<*s<<' ';
    //for(;*p!='\0';++p)
    //cout<<*p<<' ';
}

// Here there is a error in the code, as we know that the array name points to the first index of the array. What is happeining
// in the code is actually not according to the rule. We are directly using the address name to print out the whole array. What
// i have done is created another pointer and pointed on this char array and by that i have printed the whole array.
```

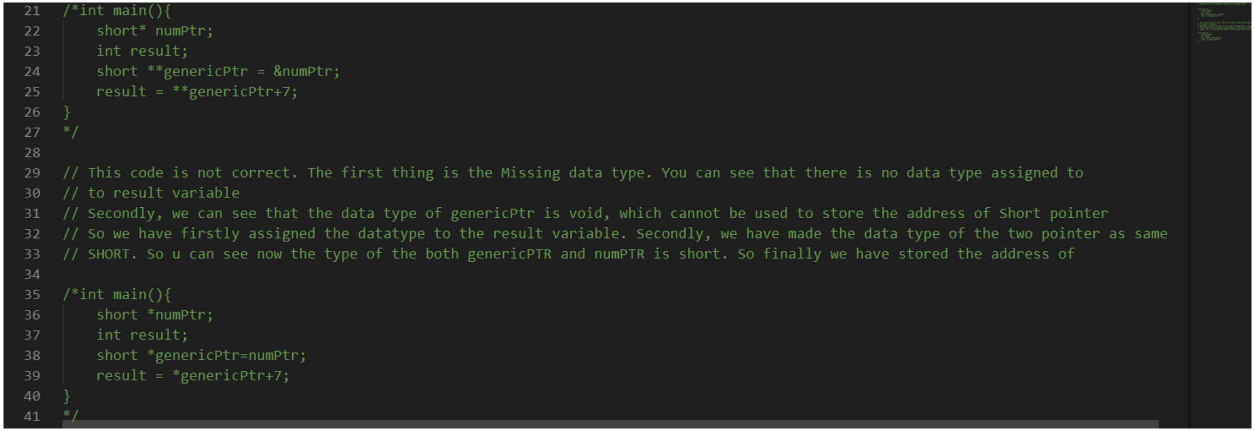


```cpp
int main(){
    char s[]="this is a character array";
    char *p;
    p = s;
    //p= p+3;
    for(;*p!='\0';++p)
    cout<<*p<<' ';
}
```

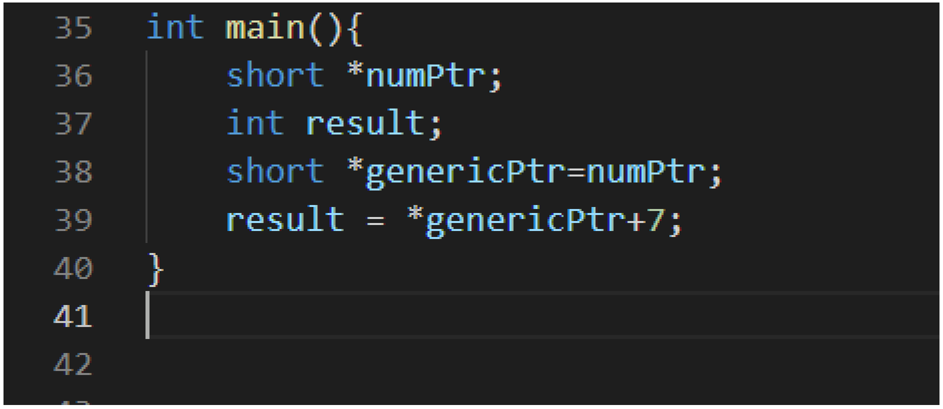
# E:



```
#include<iostream>
using namespace std;
#include<iostream>
using namespace std;
int main(){
    short *numPtr,
    result;
    void *genericPtr = numPtr;
    result = *genericPtr + 7;
}

// This code has two solutions
// Firstly, we can see that no datatype is assigned to result. Secondly, we can see that the datatypes of genericPTR and numPTR are
// different, one is void and second one is short. We have changed the datatype of genericPTR to short. Also, in order to store
// the address of a pointer, we have to made a pointer using double steric **. So we have written ** to genericPtr. So we have stored
// the address of numPtr in genericPtr.
// Also in the 4th line, we have written ** to the genericPtr in order to use its value.
// These were the all mistakes present in the above code.

```

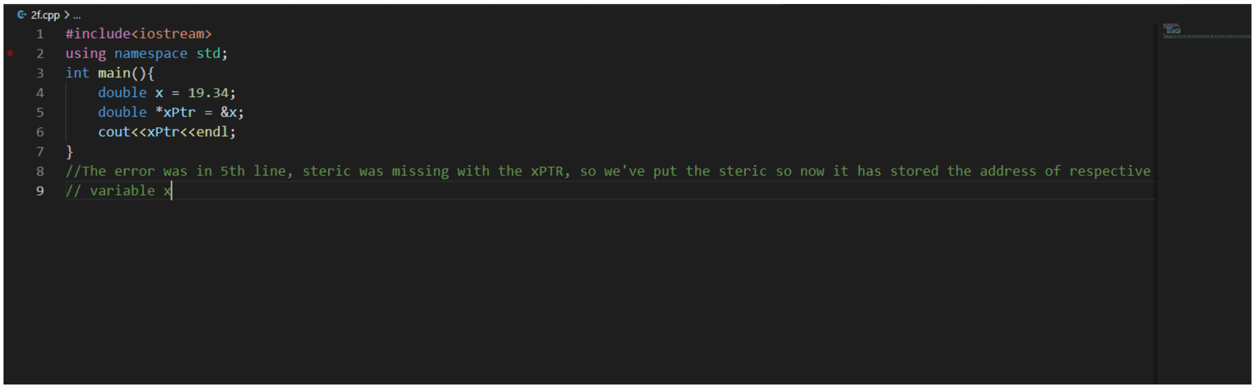


```
/*int main(){
    short* numPtr;
    int result;
    short **genericPtr = &numPtr;
    result = **genericPtr+7;
}
*/

// This code is not correct. The first thing is the Missing data type. You can see that there is no data type assigned to
// to result variable
// Secondly, we can see that the data type of genericPtr is void, which cannot be used to store the address of Short pointer
// So we have firstly assigned the datatype to the result variable. Secondly, we have made the data type of the two pointer as same
// SHORT. So u can see now the type of the both genericPTR and numPTR is short. So finally we have stored the address of

/*int main(){
    short *numPtr;
    int result;
    short *genericPtr=numPtr;
    result = *genericPtr+7;
}
*/
```

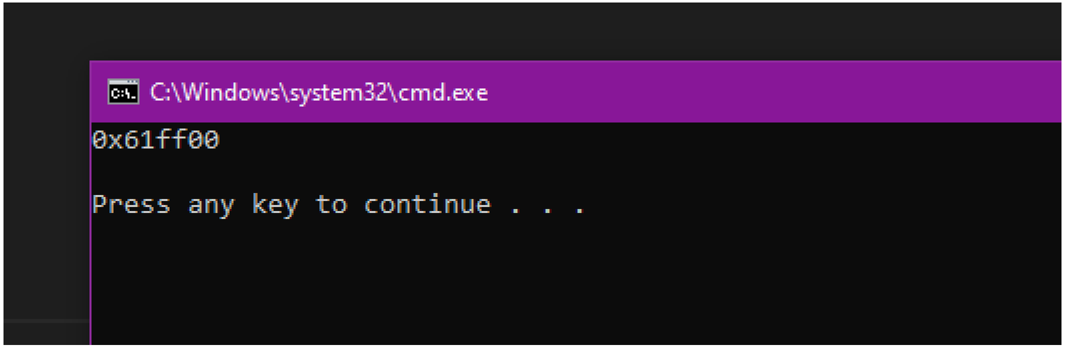


```
int main(){
    short *numPtr;
    int result;
    short *genericPtr=numPtr;
    result = *genericPtr+7;
}


```

# F:



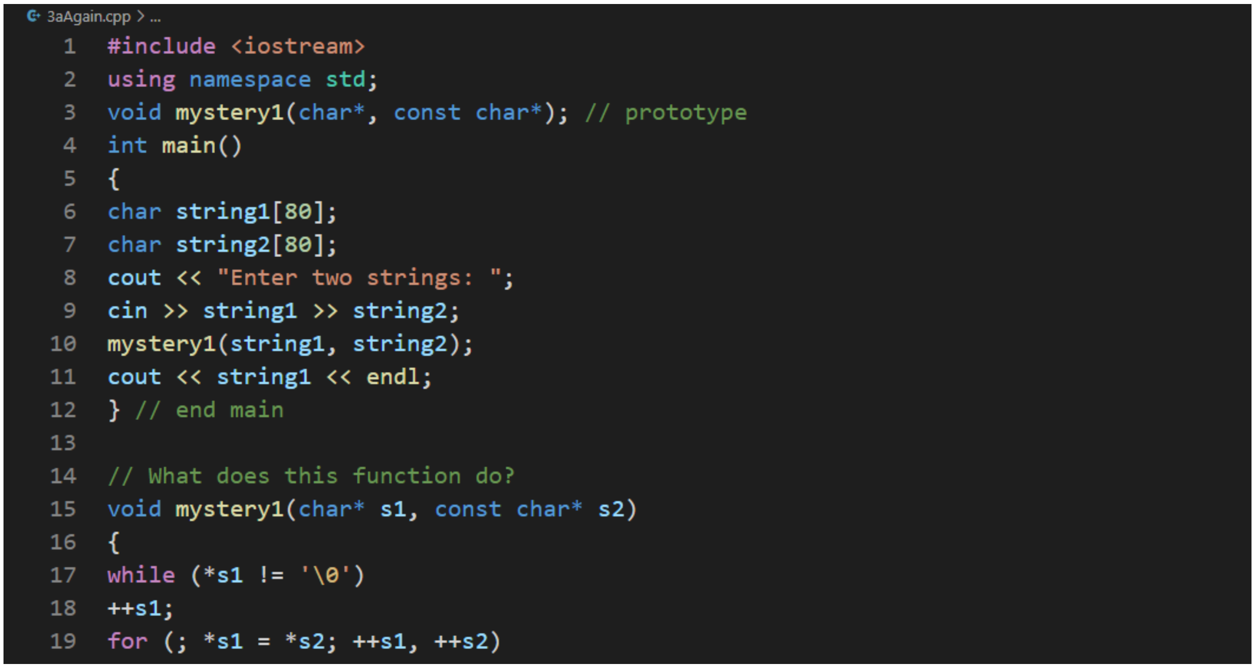
```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      double x = 19.34;
5      double *xPtr = &x;
6      cout<<xPtr<<endl;
7  }
8  //The error was in 5th line, steric was missing with the xPTR, so we've put the steric so now it has stored the address of respective
9  // variable x
```



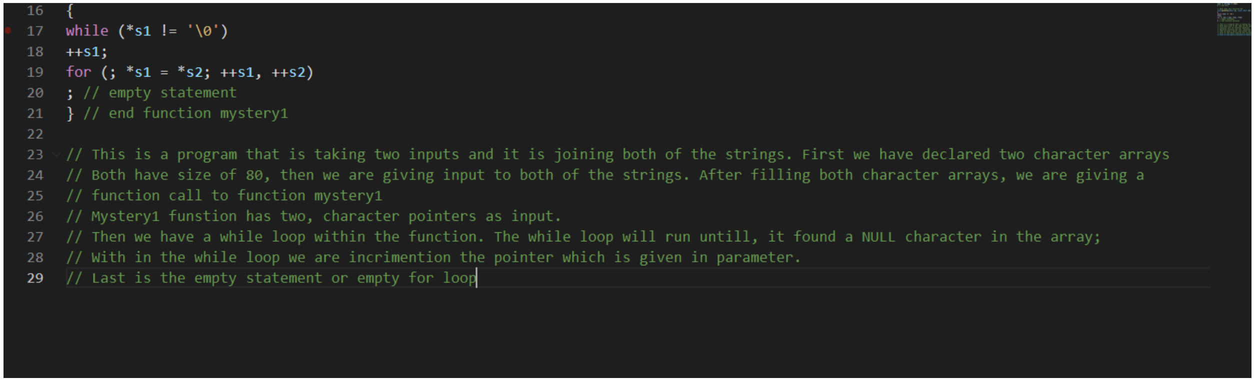
```
0x61ff00

Press any key to continue . . .
```

# Question 3:

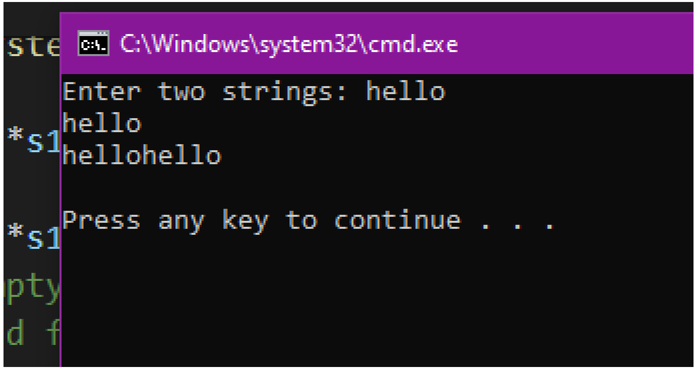
## A:



```cpp
#include <iostream>
using namespace std;
void mystery1(char*, const char*); // prototype
int main()
{
char string1[80];
char string2[80];
cout << "Enter two strings: ";
cin >> string1 >> string2;
mystery1(string1, string2);
cout << string1 << endl;
} // end main

// What does this function do?
void mystery1(char* s1, const char* s2)
{
while (*s1 != '\0')
++s1;
for (; *s1 = *s2; ++s1, ++s2)
```

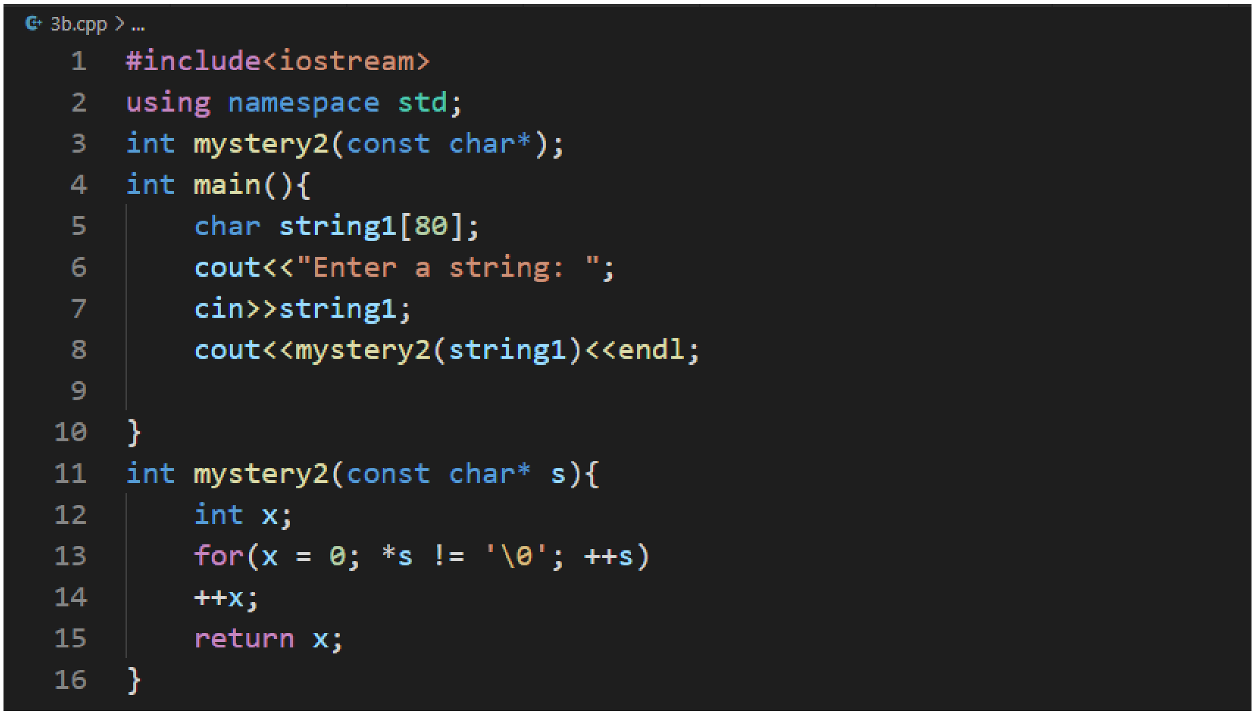


```cpp
{
while (*s1 != '\0')
++s1;
for (; *s1 = *s2; ++s1, ++s2)
; // empty statement
} // end function mystery1

// This is a program that is taking two inputs and it is joining both of the strings. First we have declared two character arrays
// Both have size of 80, then we are giving input to both of the strings. After filling both character arrays, we are giving a
// function call to function mystery1
// Mystery1 funstion has two, character pointers as input.
// Then we have a while loop within the function. The while loop will run untill, it found a NULL character in the array;
// With in the while loop we are incrimention the pointer which is given in parameter.
// Last is the empty statement or empty for loop
```

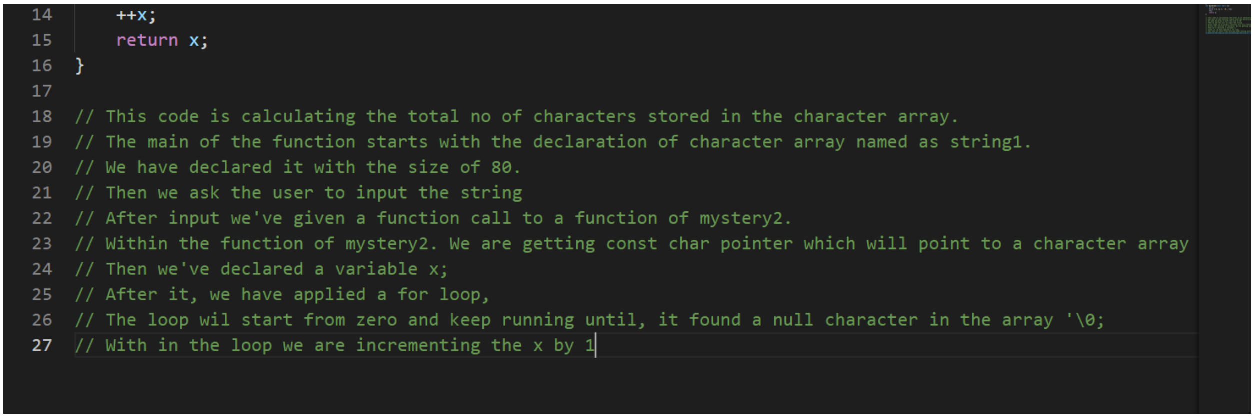


```
Enter two strings: hello
hello
hellohello

Press any key to continue . . .
```

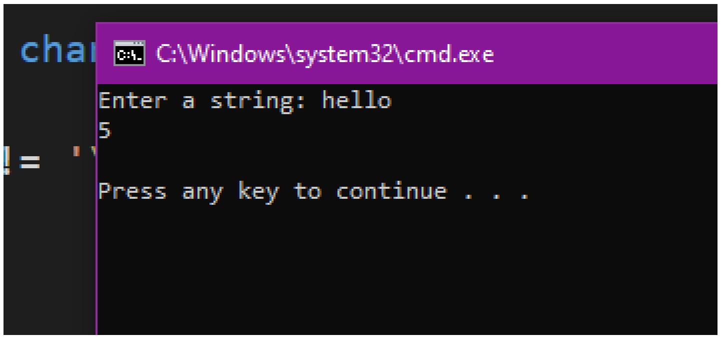
# B:



```cpp
#include<iostream>
using namespace std;
int mystery2(const char*);
int main(){
    char string1[80];
    cout<<"Enter a string: ";
    cin>>string1;
    cout<<mystery2(string1)<<endl;

}
int mystery2(const char* s){
    int x;
    for(x = 0; *s != '\0'; ++s)
    ++x;
    return x;
}
```

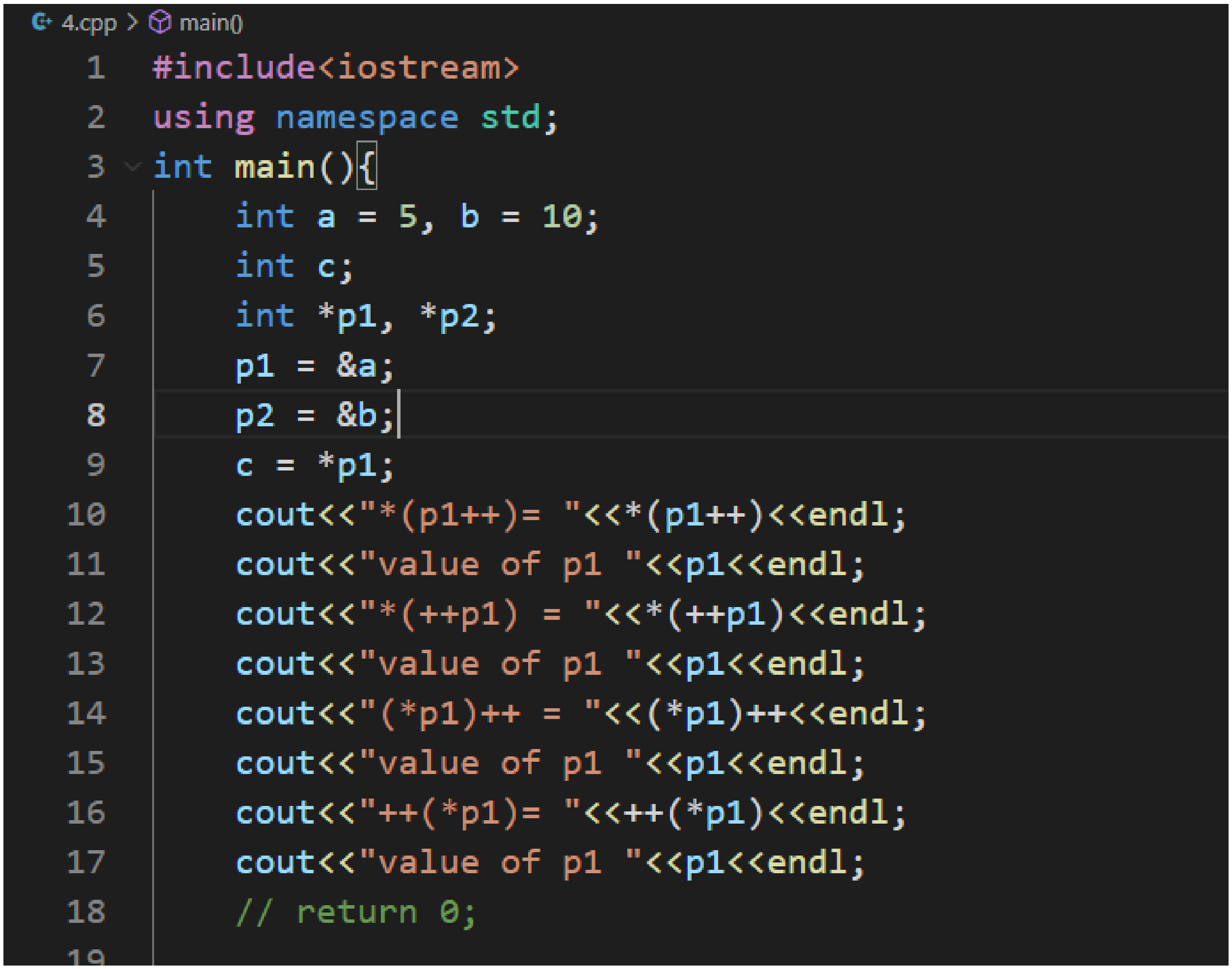


```cpp
    ++x;
    return x;
}

// This code is calculating the total no of characters stored in the character array.
// The main of the function starts with the declaration of character array named as string1.
// We have declared it with the size of 80.
// Then we ask the user to input the string
// After input we've given a function call to a function of mystery2.
// Within the function of mystery2. We are getting const char pointer which will point to a character array
// Then we've declared a variable x;
// After it, we have applied a for loop,
// The loop wil start from zero and keep running until, it found a null character in the array '\0;
// With in the loop we are incrementing the x by 1
```

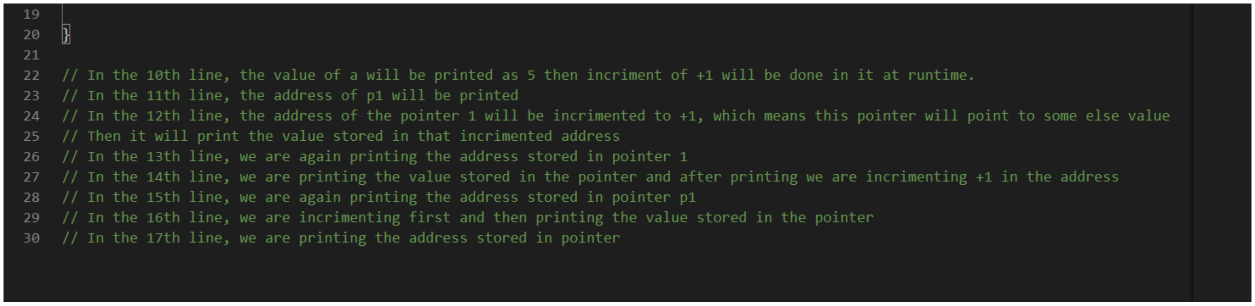


```
Enter a string: hello
5

Press any key to continue . . .
```

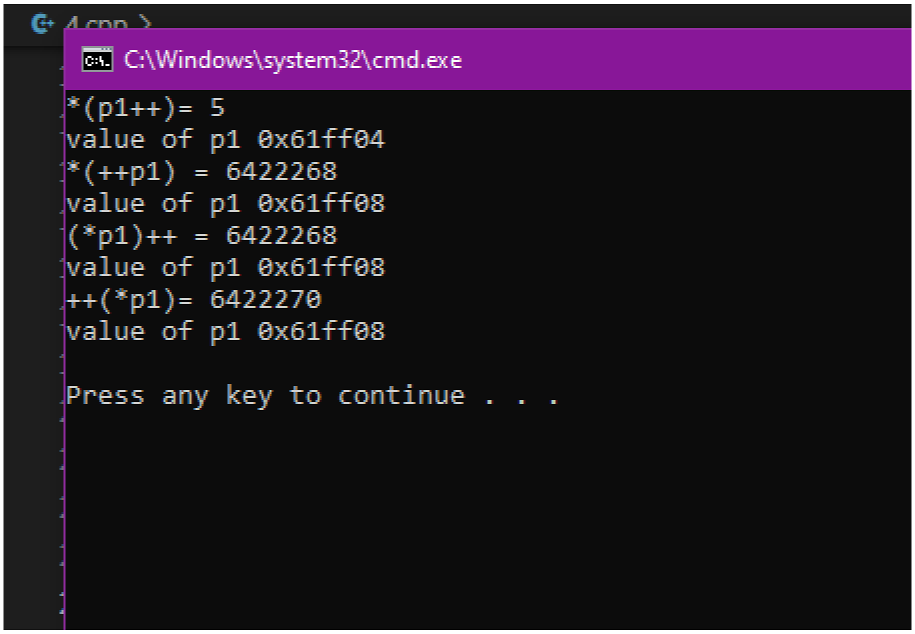
# Question 4:



```
 1  #include<iostream>
 2  using namespace std;
 3  int main(){
 4      int a = 5, b = 10;
 5      int c;
 6      int *p1, *p2;
 7      p1 = &a;
 8      p2 = &b;
 9      c = *p1;
10      cout<<"*(p1++)= "<<*(p1++)<<endl;
11      cout<<"value of p1 "<<p1<<endl;
12      cout<<"*(++p1) = "<<*(++p1)<<endl;
13      cout<<"value of p1 "<<p1<<endl;
14      cout<<"(*p1)++ = "<<(*p1)++<<endl;
15      cout<<"value of p1 "<<p1<<endl;
16      cout<<"++(*p1)= "<<++(*p1)<<endl;
17      cout<<"value of p1 "<<p1<<endl;
18      // return 0;
19
```

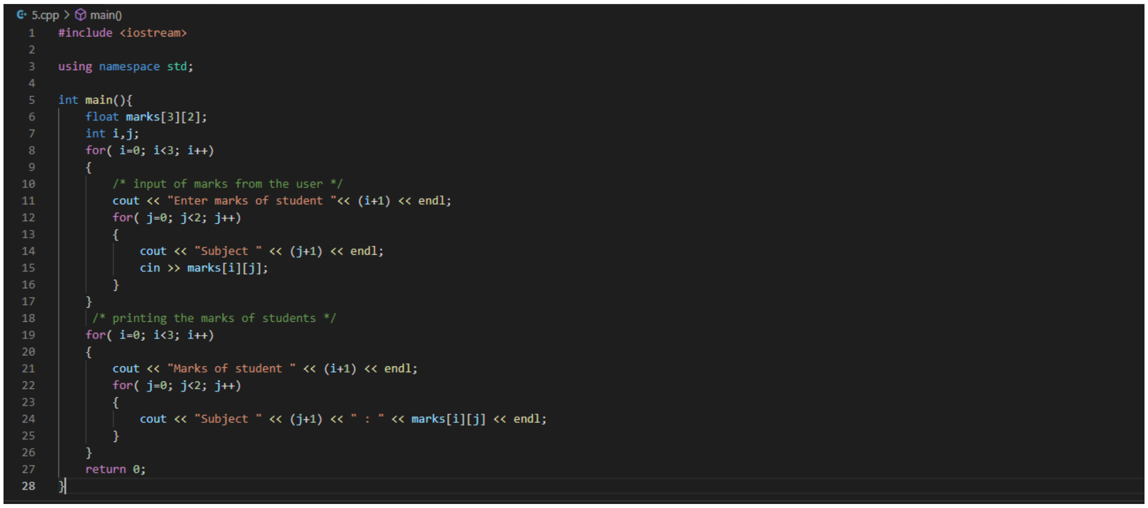


```
19
20  }
21
22  // In the 10th line, the value of a will be printed as 5 then incriment of +1 will be done in it at runtime.
23  // In the 11th line, the address of p1 will be printed
24  // In the 12th line, the address of the pointer 1 will be incrimented to +1, which means this pointer will point to some else value
25  // Then it will print the value stored in that incrimented address
26  // In the 13th line, we are again printing the address stored in pointer 1
27  // In the 14th line, we are printing the value stored in the pointer and after printing we are incrimenting +1 in the address
28  // In the 15th line, we are again printing the address stored in pointer p1
29  // In the 16th line, we are incrimenting first and then printing the value stored in the pointer
30  // In the 17th line, we are printing the address stored in pointer
```



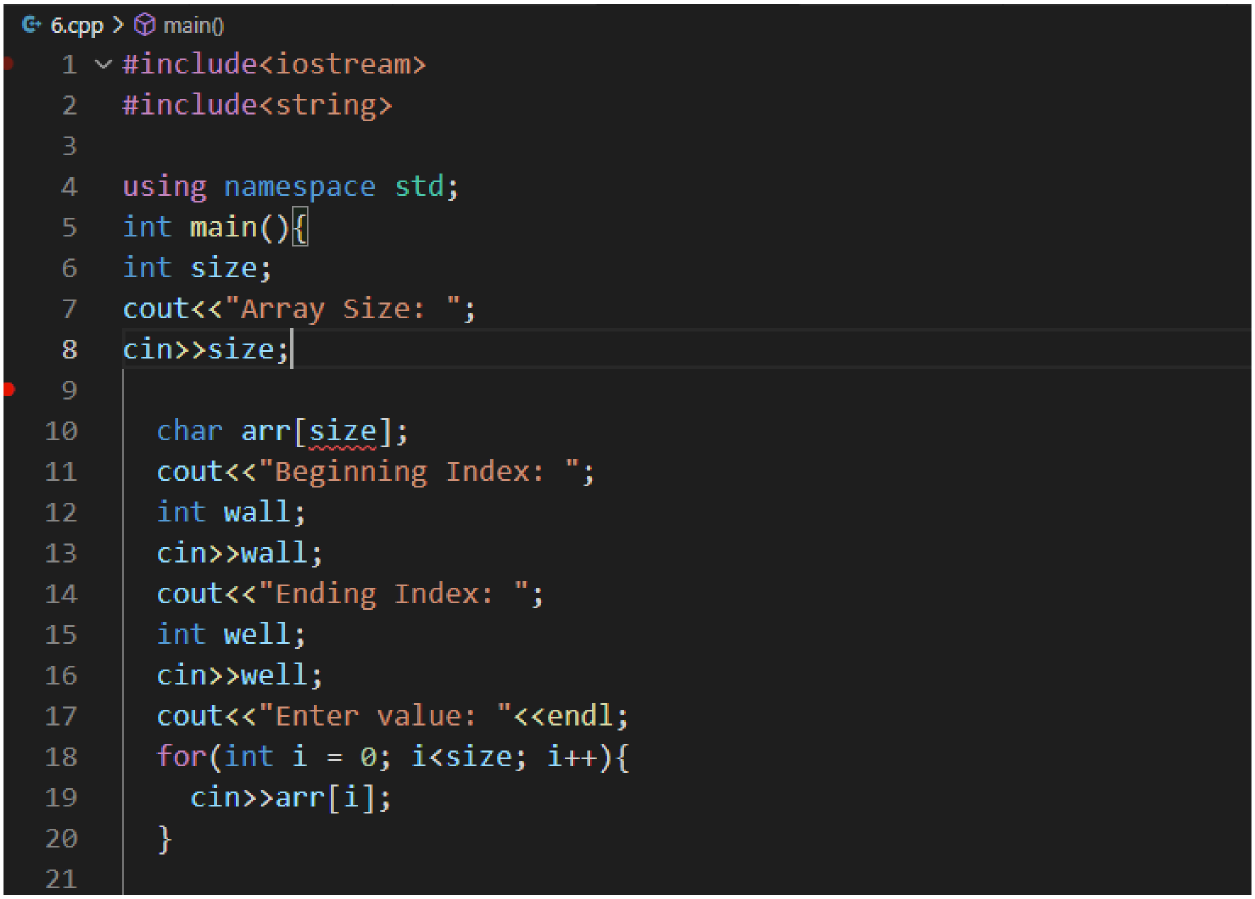
```
*(p1++)= 5
value of p1 0x61ff04
*(++p1) = 6422268
value of p1 0x61ff08
(*p1)++ = 6422268
value of p1 0x61ff08
++(*p1)= 6422270
value of p1 0x61ff08

Press any key to continue . . .
```

# Question 5:

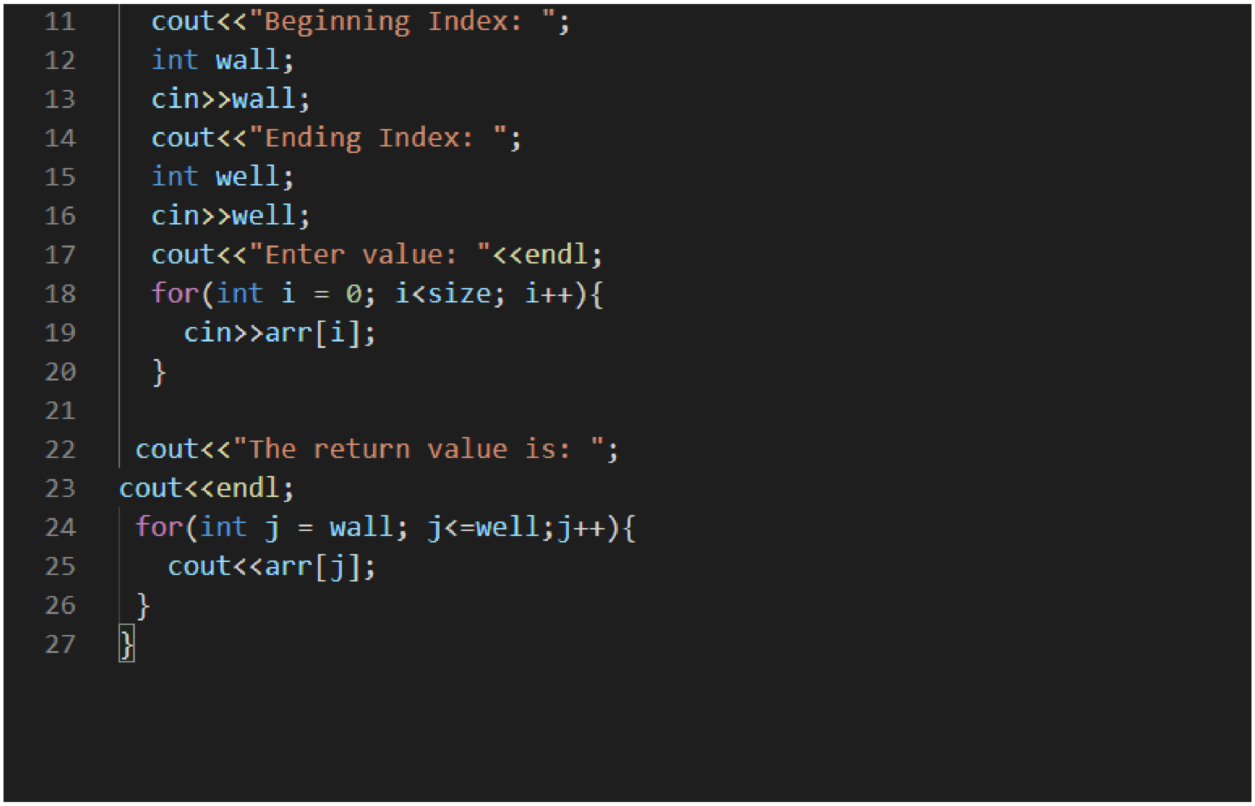


```cpp
#include <iostream>

using namespace std;

int main(){
    float marks[3][2];
    int i,j;
    for( i=0; i<3; i++)
    {
        /* input of marks from the user */
        cout << "Enter marks of student "<< (i+1) << endl;
        for( j=0; j<2; j++)
        {
            cout << "Subject " << (j+1) << endl;
            cin >> marks[i][j];
        }
    }
      /* printing the marks of students */
    for( i=0; i<3; i++)
    {
        cout << "Marks of student " << (i+1) << endl;
        for( j=0; j<2; j++)
        {
            cout << "Subject " << (j+1) << " : " << marks[i][j] << endl;
        }
    }
    return 0;
}
```

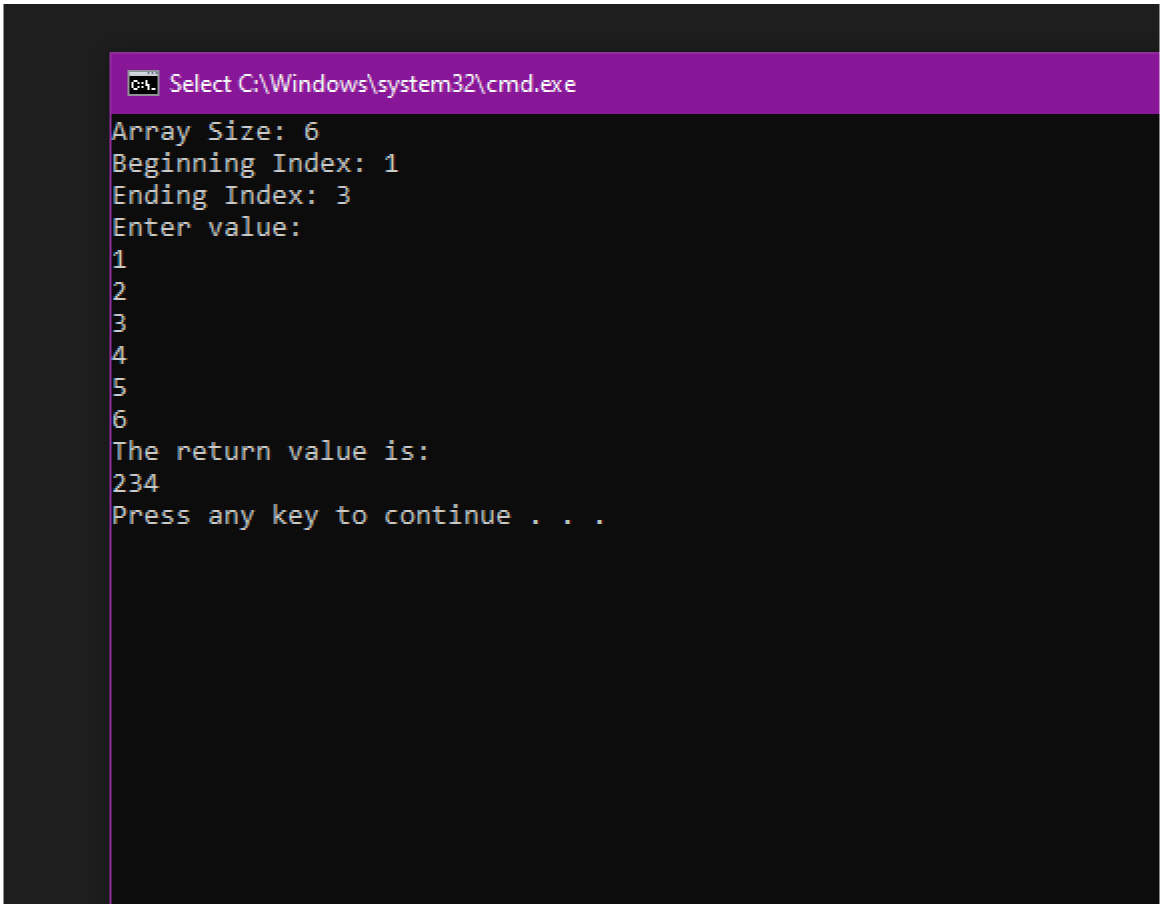
# Question 6:



```
#include<iostream>
#include<string>

using namespace std;
int main(){
int size;
cout<<"Array Size: ";
cin>>size;

  char arr[size];
  cout<<"Beginning Index: ";
  int wall;
  cin>>wall;
  cout<<"Ending Index: ";
  int well;
  cin>>well;
  cout<<"Enter value: "<<endl;
  for(int i = 0; i<size; i++){
    cin>>arr[i];
  }

```



```
  cout<<"Beginning Index: ";
  int wall;
  cin>>wall;
  cout<<"Ending Index: ";
  int well;
  cin>>well;
  cout<<"Enter value: "<<endl;
  for(int i = 0; i<size; i++){
    cin>>arr[i];
  }

 cout<<"The return value is: ";
cout<<endl;
 for(int j = wall; j<=well;j++){
   cout<<arr[j];
 }
}
```



```
Array Size: 6
Beginning Index: 1
Ending Index: 3
Enter value:
1
2
3
4
5
6
The return value is:
234
Press any key to continue . . .
```