

PPL CLASS TUTORIAL

NAME: SOURABH PATEL

ADMISSION NO: U19CS082

Q. 1. Write a program that requests the user to enter two integers. The program should then calculate and report the sum of all the integers between and including the two integers. At this point, assume that the smaller integer is entered first. For example, if the user enters 2 and 9, the program should report that the sum of all the integers from 2 through 9 is 44.

```
#include <iostream>
using namespace std;

int main(){
    int a,b; int sum=0;
    cout<<"Enter smaller number: "; cin>>a;
    cout<<"Enter larger number: "; cin>>b;
    if(a>b){
        cout<<"Invalid inputs.";
    }
    else{
        for(int i=a;i<=b;i++){
            sum+=i;
        }
        cout<<"Sum = "<<sum;
    }
    return 0;
}
```

```
C:\Users\Sourabh Patel\Desktop\assignment\82\SEM6\PPL\CLASSTEST\Q1.exe
Enter smaller number: 6
Enter larger number: 9
Sum = 30
-----
Process exited after 8.929 seconds with return value 0
Press any key to continue . . .
```

Q2. Write a program that opens a text file, reads it character-by-character to the end of the file, and reports the number of characters in the file.

```
#include <iostream>
#include <fstream>
using namespace std;

int main(){
    int cnt=0; char ch;
    ifstream file("file.txt");
    if (!file.is_open()) {
        cerr << "Could not open the file - '" << "file.txt" << "'" << endl;
        return EXIT_FAILURE;
    }

    while (file.get(ch)) {
        //cout<<ch<<'-';
        cnt++;
    }
    cout<<"Character count = "<<cnt; return 0;
}
```

file.txt

1 Welcome to SVNIT Surat

C:\Users\Sourabh Patel\Desktop\assignment\82\SEM6\PPL\CLASSTEST\Q2.exe

Character count = 22

Process exited after 0.1106 seconds with return value 0

Press any key to continue . . .

Q3. Write a program that reads up to 10 donation values into an array of double. (Or, if you prefer, use an array template object). The program should terminate input on non-numeric input. It should report the average of the numbers and also report how many numbers in the array are larger than the average.

```
#include <iostream>
using namespace std;

#define SIZE 10

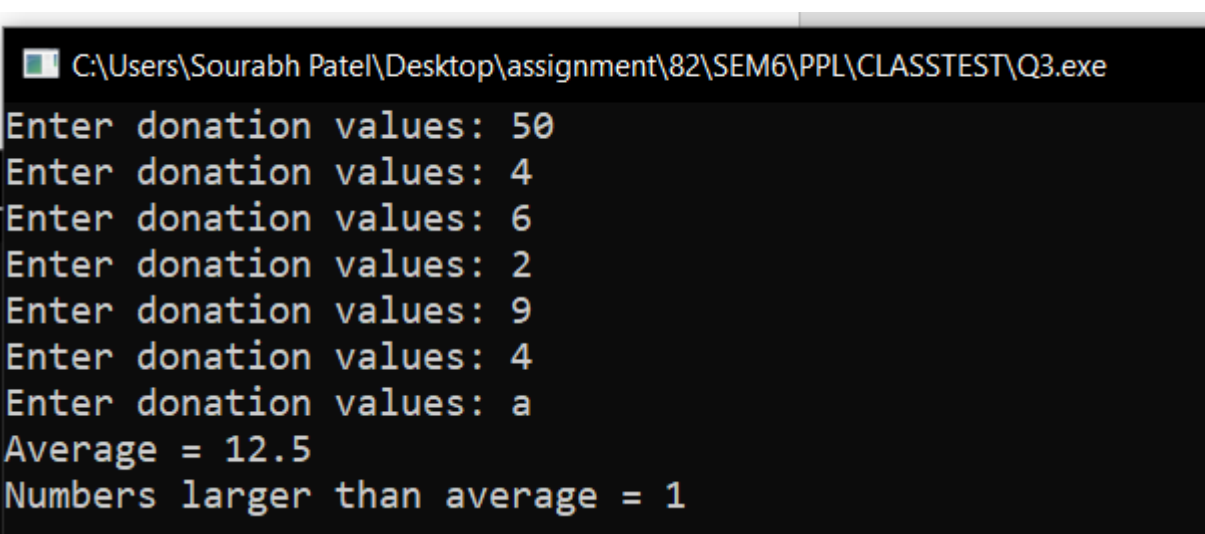
int main(){
    double arr[SIZE];
    double donation_val, sum = 0.0; int cnt = 0, i = 0;
```

```

cout << "Enter donation values: ";
for(; i < SIZE && cin >> donation_val; i++){
    arr[i]=donation_val;
    sum+=donation_val;
    cout<<"Enter donation values: ";
}
if (i != 0){
    double avg = sum/i;
    for(int j = 0; j < i; j++){
        if (arr[j] > avg)
            cnt++;
    }
    cout<<"Average = "<<avg<<"\n";
    cout<<"Numbers larger than average = "<<cnt<<"\n";
}
else
    cout<<"No data entered."<<endl;

return 0;
}

```



```

C:\Users\Sourabh Patel\Desktop\assignment\82\SEM6\PPL\CLASSTEST\Q3.exe
Enter donation values: 50
Enter donation values: 4
Enter donation values: 6
Enter donation values: 2
Enter donation values: 9
Enter donation values: 4
Enter donation values: a
Average = 12.5
Numbers larger than average = 1

```

Q4. a. Write a function that passes a box structure by value and that displays the value of each member.

- b. Write a function that passes the address of a box structure and that sets the volume member to the product of the other three dimensions.
- c. Write a simple program that uses these two functions.
- d. Write a function that has a reference to a box structure as its formal argument and displays the value of each member (Separate program point d & e).
- e. Write a function that has a reference to a box structure as its formal argument and sets the volume member to the product of the other three dimensions.

```
#include <iostream>
using namespace std;

struct box{
    char maker[40];
    float height;
    float width;
    float length;
    float volume;
};

//(a)
void show_box(const box t){
    cout << "Maker: " << t.maker << "\nHeight: " << t.height
    << "\nWidth: " << t.width << "\nLength: " << t.length
    << "\nVolume: " << t.volume << endl;
}

//(b)
void volume_box(box * t){
    t->volume = t->height * t->width * t->length;
}

//(d)
void show_box_by_ref(const box &t){
    cout << "Maker: " << t.maker << "\nHeight: " << t.height
    << "\nWidth: " << t.width << "\nLength: " << t.length
    << "\nVolume: " << t.volume << endl;
}

//(e)
void volume_box_by_ref(box &t){
    t.volume = t.height * t.width * t.length;
}

//(c)
int main(){

    printf("First cube:- ");
    box cube = {"First Cube", 5, 4, 10, 0};
    //show_box(cube);
    volume_box(&cube);
    show_box(cube);
}
```

```

printf("\nSecond cube:- ");
box cube1 = {"Second cube", 5, 4, 10, 0};
//show_box(cube1);
volume_box_by_ref(cube1);
show_box_by_ref(cube1);
return 0;
}

```

```

C:\Users\Sourabh Patel\Desktop\assignment\82\SEM6\PPL\CLASSTEST\Q4.exe
First cube:- Maker: First Cube
Height: 5
Width: 4
Length: 10
Volume: 200

Second cube:- Maker: Second cube
Height: 5
Width: 4
Length: 10
Volume: 200

-----
Process exited after 0.09951 seconds with return value 0
Press any key to continue . . .

```

Q5. Fill_array() takes as arguments the name of an array of double values and an array size. It prompts the user to enter double values to be entered in the array. It ceases taking input when the array is full or when the user enters non-numeric input, and it returns the actual number of entries.

Show_array() takes as arguments the name of an array of double values and an array size and displays the contents of the array.

Reverse_array() takes as arguments the name of an array of double values and an array size and reverses the order of the values stored in the array.

The program should use these functions to fill an array, show the array, reverse the array, show the array, reverse all but the first and last elements of the array, and then show the array.

```

#include<bits/stdc++.h>
using namespace std;
#define ARR_SIZE 10

int Fill_array(double arr[], int size){
    memset(arr,0,size);
}

```

```

    cout << "You may enter up to " << size << " values." << endl; cout << "Enter value number 1 (or
type \"q\" when finished): ";
    int count = 0;
    while (count < size && cin >> arr[count])
    {
        if (++count < size)
            cout << "Enter value number " << count + 1 << " (or type \"q\" when finished): ";
    }
    return count;
}

void Show_array(const double arr[], int size){
    cout << endl;
    for (int i = 0; i < size; i++)
        cout << "Element number " << i + 1 << " = " << arr[i] << endl;
    cout << endl;
    return;
}

void Reverse_array(double arr[], int size){
    int temp;
    for (int i = 0; i < size / 2; i++)
    {
        temp = arr[i];
        arr[i] = arr[(size - 1) - i];
        arr[(size - 1) - i] = temp;
    }
    return;
}

int main(void){
    double arr[ARR_SIZE];
    int entries = Fill_array(arr, ARR_SIZE);

    // show, reverse, show, reverse all but first/last elements, show
    cout << endl << "Here is the
array:" << endl;
    Show_array(arr, entries);
    cout << "Let's reverse the array." << endl;
    Reverse_array(arr, entries);
    cout << "Here is our new array:" << endl;
    Show_array(arr, entries);
    cout << "Let's reverse all except the first and last elements." << endl;

    Reverse_array(arr + 1, entries - 2);
    cout << "Here is our final array:" << endl;
    Show_array(arr, entries);

    return 0;
}

```

```

You may enter up to 10 values.
Enter value number 1 (or type "q" when finished): 10
Enter value number 2 (or type "q" when finished): 11
Enter value number 3 (or type "q" when finished): 12
Enter value number 4 (or type "q" when finished): 13
Enter value number 5 (or type "q" when finished): 14
Enter value number 6 (or type "q" when finished): 15
Enter value number 7 (or type "q" when finished): 16
Enter value number 8 (or type "q" when finished): 17
Enter value number 9 (or type "q" when finished): 18
Enter value number 10 (or type "q" when finished): q

Here is the array:

Element number 1 = 10
Element number 2 = 11
Element number 3 = 12
Element number 4 = 13
Element number 5 = 14
Element number 6 = 15
Element number 7 = 16
Element number 8 = 17
Element number 9 = 18

Let's reverse the array.
Here is our new array:

Element number 1 = 18
Element number 2 = 17
Element number 3 = 16
Element number 4 = 15
Element number 5 = 14
Element number 6 = 13
Element number 7 = 12
Element number 8 = 11
Element number 9 = 10

Let's reverse all except the first and last elements.
Here is our final array:

Element number 1 = 18
Element number 2 = 11
Element number 3 = 12
Element number 4 = 13
Element number 5 = 14
Element number 6 = 15
Element number 7 = 16
Element number 8 = 17
Element number 9 = 10

Process returned 0 (0x0)   execution time : 15.294 s

```

Q.6. Suppose the song() function has this prototype:

```
void song(const char * name, int times);
```

- How would you modify the prototype so that the default value for times is 1?
- What changes would you make in the function definition?
- Can you provide a default value of "O, My Papa" for the name?

PPL Programming Practice Question

⑥ (a) void song(const char* name, int times = 1);
 giving the default value

⑦ (b) This is a prototype

(Prototype :- there are refer to create a copy of an object without knowing its concrete type. Hence it is also known as "virtual copy constructor".)

→ No need to change in the function definition, only prototypes contain the default value info.

⑧ (c) Yes, void song(const char* name = "O, my Papa",
 int times = 1);

Q7. Which of the following initializations are legal? Explain why.

- (a) int i = -1, &r = 0;
- (b) int *const p2 = &i2;
- (c) const int i = -1, &r = 0;
- (d) const int *const p3 = &i2;
- (e) const int *p1 = &i2;
- (f) const int &const r2;
- (g) const int i2 = i, &r = i;

⑦ (a) int i = -1, &r = 0;
 • Invalid

→ Here the "Initializer" to the reference & must be an object, or a constant value.

⑧ (b) int *const p2 = &i2
 → It is legal

- (c) `const int r = -1, &M = 0;`
 \Rightarrow It is valid to reference 'M' as it is assigned to constant value.
- (d) `const int & const p3 = &p2`
 \Rightarrow It is legal.
- (e) `const int & p1 = &p2;`
 \Rightarrow It is legal.
- (f) `const int & const M2;`

Here M2 requires the "initialization"

- (g) `const int p2 = r, &M = 1;`
 \Rightarrow It is legal.

Q8. Explain the following definitions. Identify any that are illegal.

- (a) `int i, *const cp;` (b) `int *p1, *const p2;` (c) `const int ic, &r = ic;`
 (d) `const int *const p3;` (e) `const int *p;`

- (a) `int i, *const cp;`
 \Rightarrow Illegal, `int *const cp` is illegal because `const` should be initialized while declaring.
- (b) `int *p1, *const p2;`
 \Rightarrow Illegal, `int *const p2` is illegal because "const" should be initialized while declaring.
- (c) `const int ic, &r = ic;`
 \Rightarrow Illegal, `const int ic` is uninitialized.
- (d) `const int *const p3;`
 \Rightarrow Illegal, `const int *const p3;` is initialized.
- (e) `const int *p;`
 \Rightarrow Legal, `p` is const integer type pointer.

Q9. Using the variables in the previous exercise, which of the following assignments

are legal? Explain why.

(a) `i = ic;` (b) `p1 = p3;`

(c) `p1 = ⁣` (d) `p3 = ⁣`

(e) `p2 = p1;` (f) `ic = *p3;`

(a) `p = ic;`
legal, Assuming that `p` is initialized then the above assignment are legal.

(b) `p1 = p3;`
Illegal, Invalid conversion from "const int*" (`p3`) to "int".

(c) `p1 = ⁣`
Illegal, Invalid conversion from "const int*" (`ic`) to "int*".

(d) `p3 = ⁣`

Illegal, Assignment of `p3` is not allowed because type of `p3` is constant.

(e) `p2 = p1;`
Illegal, `p2` is of constant type (read-only).

(f) `ic = *p3;`
Illegal, `ic` is of constant type (read-only).

Q10. Assuming `txt_size` is a function that takes no arguments and returns an `int` value, which of the following definitions are illegal? Explain why.

`unsigned buf_size = 1024;`

(a) `int ia[buf_size];` (b) `int ia[4 * 7 - 14];`

(c) `int ia[txt_size()];` (d) `char st[11] = "fundamental";`

10 (a) `int ia (buf-size);`
legal.

(b) `int ia ($ * = 10);`
⇒ legal

(c) `int ia (+x+size);`
legal

(d) `char st[11] = "fundamental";`

⇒ illegal, because of size is not enough to store the string.

Q11. Correct the errors in each of the following code fragments:

(a) `if (ival1 != ival2)`

`ival1 = ival2`

`else ival1 = ival2 = 0;`

(b) `if (ival < minval)`

`minval = ival;`

`occurs = 1;`

(c) `if (int ival = get_value())`

`cout << "ival = " << ival << endl;`

`if (!ival)`

`cout << "ival = 0\n";`

(d) `if (ival = 0)`

`ival = get_value();`

11 (a) `if (ival1 != ival2)`

`ival1 = ival2`

`else ival1 = ival2 = 0;`

⇒ semicolon is needed after `ival1 = ival2`.

① if (ival < min_val)
 min_val = ival;
 occurs = 1;

⇒ 2) "min_val = ival" + "occurs = 1" are within the
 if cond. scope then parenthesis "if" & "4" is misplaced

② i) cin >> ival = get_value();
 cout << "ival = " << ival << endl;
 if (ival)
 cout << "ival = 0\n";

⇒ second if condition should be else if

③ if (ival == 0)
 ival = get_value();
 ⇒ ival == 0 should be there in if condition

Q12. Explain each of the following loops. Correct any problems you detect.

(a) do
 int v1, v2;
 cout << "Please enter two numbers to sum:" ;
 if (cin >> v1 >> v2)
 cout << "Sum is: " << v1 + v2 << endl;
 while (cin);

(b) do { // ... } while (int ival = get_response());

(c) do { int ival = get_response(); } while (ival);

④ a) Another is missing for "do" part of do while loop
 do {
 int v1, v2;
 cout << "please enter two numbers to sum:";
 if (cin >> v1 >> v2)
 cout << "sum is: " << v1 + v2 << endl;
 }
 while (cin);

(b) `flag` should not be declared inside the while condition.

```
int flag;
do {
    // ...
} while(flag = get_response());
```

(c) `flag` is outside the scope of while, so `flag` should be declared before do.

```
=> int flag;
do {
    flag = get_response();
} while(flag);
```

Q13. Indicate which of the following functions are in error and why. Suggest how you might correct the problems.

(a) `int f() {`
`string s;`
`// ...`
`return s;`
`}`

(b) `f2(int i) { /* ... */ }`

(c) `int calc(int v1, int v1) /* ... */ }`

(d) `double square(double x) return x * x;`

(a) => error, return type of `f` is `int`, but we are trying to return string.

(b) => error, return type is not written for the fun.

(c) => error, duplicate parameters, name of parameter is also missing.

Q14. Define a pair of classes X and Y, in which X has a pointer to Y, and Y has an object of type X.

```
(14) #include <iostream>
using namespace std;
class Y;
class X {
    Y * y;
};
class Y {
    X * x;
};
int main() {
    return 0;
}
```

Q15. Which, if any, of the following statements are untrue? Why?

- (a) A class must provide at least one constructor.
- (b) A default constructor is a constructor with an empty parameter list.
- (c) If there are no meaningful default values for a class, the class should not provide a default constructor.
- (d) If a class does not define a default constructor, the compiler generates one that initializes each data member to the default value of its associated type.

15. (a) True, if the programmer doesn't provide any constructor, the compiler will synthesize one default constructor.

(b) A default constructor is a constructor with empty parameter list.

→ Unlike. A constructor of which all parameters have default value which define a default constructor.

(c) Unlike. the user should provide.

(d) Unlike. only if user doesn't explicitly define any constructor, the compiler will implicitly define the default constructor for it.

Q16. The CandyBar structure contains three members. The first member holds the brand name of a candy bar. The second member holds the weight (which may have a fractional part) of the candy bar, and the third member holds the number of calories (an integer value) in the candy bar. Write a program that declares such a structure and creates a CandyBar variable called snack, initializing its members to "Mocha Munch", 2.3, and 350, respectively. The initialization should be part of the declaration for snack. Finally, the program should display the contents of the snack variable.

```
#include <iostream>
using namespace std;
struct CandyBar
{
    string brand; float weight; int cal;
};

int main()
{
    CandyBar snack { "Mocha Munch", 2.3, 350 };
    cout << "Struct snack\n Brand name: " << snack.brand << "\n Weight: "
    << snack.weight << "\n Calories: " << snack.cal << std::endl;
    return 0;
}
```

```
Struct snack
Brand name: Mocha Munch
Weight: 2.3
Calories: 350

Process returned 0 (0x0)   execution time : 0.227 s
Press any key to continue.
```

Q17. Write a program that uses an array of char and a loop to read one word at a time until the word done is entered. The program should then report the number of words entered (not counting done). A sample run could look like this:

Enter words (to stop, type the word done):

anteater birthday category dumpster

envy finagle geometry done for sure

You entered a total of 7 words.

You should include the cstring header file and use the strcmp() function to make the comparison test.

```
#include <iostream>
#include <cstring>

int main()
{
    char word[20];
    int count = 0;
    std::cout << "Enter words (to stop, type the word done):\n";
    std::cin >> word;
    while (strcmp(word, "done"))
    {
        std::cin >> word; count++;
    }
    std::cout << "You entered a total of " << count << " words." << "\n";
    return 0;
}
```



```
Enter words (to stop, type the word done):
anteater birthday category dumpster
envy finagle geometry done for sure
You entered a total of 7 words.

Process returned 0 (0x0)   execution time : 0.800 s
Press any key to continue.
```

Q18. Carefully consider the following program:

```
#include <iostream>

using namespace std;

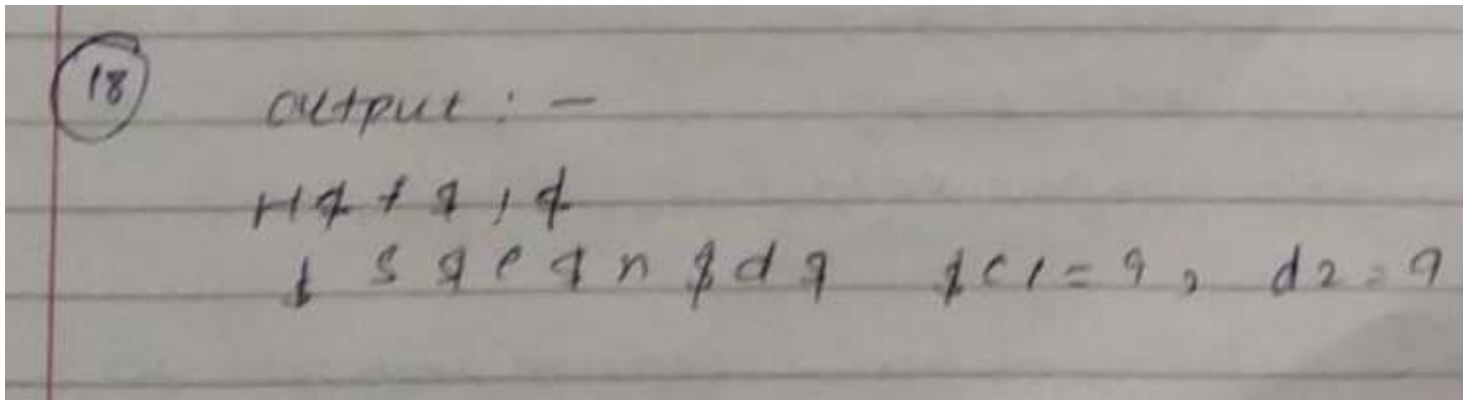
int main(){
    char ch;
    int ct1, ct2;
    ct1 = ct2 = 0;
    while ((ch = cin.get()) != '$'){
        cout << ch;
        Ct1++;
        if (ch = '$')
            Ct2++;
        cout << ch;
    }
    cout << "ct1 = " << ct1 << ", ct2 = " << ct2 << "\n";
    return 0;
}
```

Suppose you provide the following input, pressing the Enter key at the end of each line:

Hi!

Send \$10 or \$20 now!

What is the output?



Q19. Consider the following skeletal C program:

```
void fun1(void); /* prototype */
void fun2(void); /* prototype */
void fun3(void); /* prototype */

void main() {
    int a, b, c;
    ...}

void fun1(void) {
    int b, c, d;
    ...}

void fun2(void) {
    int c, d, e;
    ...}

void fun3(void) {
    int d, e, f;
    ...}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.

- main calls fun1; fun1 calls fun2; fun2 calls fun3.
- main calls fun1; fun1 calls fun3.
- main calls fun2; fun2 calls fun3; fun3 calls fun1.
- main calls fun3; fun3 calls fun1.
- main calls fun1; fun1 calls fun3; fun3 calls fun2.
- main calls fun3; fun3 calls fun2; fun2 calls fun1.

19	5	variable	scope
		a	main
		b	fun1
		c	fun2
		d, e, f	fun3

5	variable	scope
	a	main
	b, c	fun1
	d, e, f	fun3

6	variable	scope
	a	main
	b, c, d	fun1
	e, f	fun3

7	variable	scope
	a	main
	b, c, d	fun1
	e, f	fun3

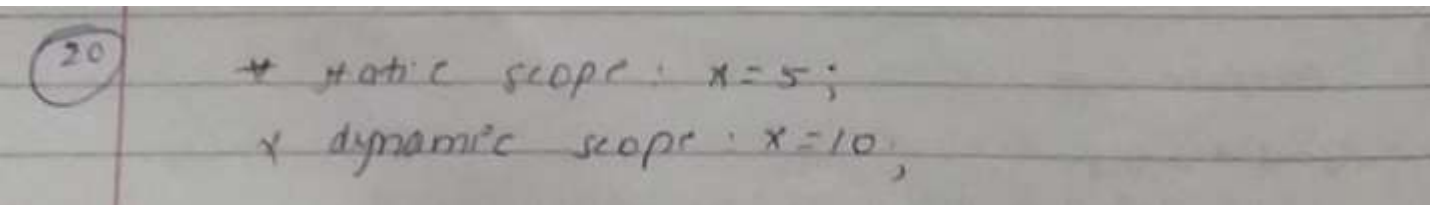
8	variable	scope
	a	main
	c, d, e	fun2
	b	fun1
	f	fun3

9	variable	scope
	a	main
	b, c, d	fun1
	f	fun3
	e	fun2

Q20. Assume the following JavaScript program was interpreted using static-scoping rules. What value of x is displayed in function sub1? Under dynamic-scoping rules, what value of x is displayed in function sub1?

```
var x;
function sub1()
{
  document.write("x = " + x + "<br />");
}
```

```
function sub2()
{
    var x;
    x = 10;
    sub1();
}
x = 5;
var x; x = 10;
sub1();
sub2();
```



Q21. Given the following classes, explain each print function:

```
class base {
public:
    string name() { return basename; }
    virtual void print(ostream &os) { os << basename; }
private:
    string basename;
};

class derived : public base {
public:
    void print(ostream &os) { print(os); os << " " << i; }
private:
    int i;
};
```

If there is a problem in this code, how would you fix it?

class base {

public:

string name(); returns basename;

virtual void print (ostream &os)

{ os << basename; }

private:

string basename;

};

class derived : public base {

public:

void print (ostream &os) { print(os); os << "d" }

private:

int r;

};

→ Here print(os) in derived class has more priority than the base class (as it is virtual fun) so that "print(os)" call call infinitely then which infinite loop error, for that we have to use a scope resolution operator to stop that

void print (ostream &os) { base::print(os); os << "d" << endl; }

reverseArray()

#include <bits/stdc++.h>

using namespace std;

class ArrayFun {

public:

int callArray (double arr[], int arr-size);

void showArray (double arr[], int arr-size);

void reverseArray (double arr[], int arr-size);

};

int ArrayFun::callArray (double arr[], int arr-size)

{

```

for (int i=0; i < arr-size; i++)
{
    double input;
    cout << "enter a number: ";
    cin >> input;
    if (isdigit(input))
    {
        arr[i] = input;
    }
    else { arr-size = i;
        return i;
    }
}
return arr-size;
}
}

```

```

void array_fun :: show_array
(double arr[], int arr-size)
{
    for (int i=0; i < arr-size; i++)
    {
        cout << arr[i] << " ";
    }
}

```

```

void array_fun :: Reverse_array
(double arr[], int arr-size)
{
    double temp[arr-size];
    for (int i = arr-size; i >= 0; i--)
        reverse[i] = arr[arr-size-i];
}
for (int i=0 to P < arr-size) <
    cout << reverse[i] << " ";
}
}

```

brand name of a candy bar. The second member holds the weight (which may have a fractional part) of the candy bar, and the third member holds the number of calories (an integer value) in the candy bar. Write a program that uses a function that takes as arguments a reference to CandyBar, a pointer-to-char, a double, and an int and uses the last three values to set the corresponding members of the structure. The last three arguments should have default values of "Millennium Munch," 2.85, and 350. Also the program should use a function that takes a reference to a CandyBar as an argument and displays the contents of the structure. Use const where appropriate.

Q23. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume Bigsub is at level 1.

```
function bigsub() {
  function a() {
    function b() {
      ... <-----1
    } // end of b
    function c() {
      ...
      b();
      ...
    } // end of c
    ...
    c();
    ...
  } // end of a
  ...
  a();
  ...
} // end of bigsub
```


Q24. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume Bigsub is at level 1.

```
function bigsub() {  
  var mysum;  
  function a() {  
    var x;  
    function b(sum) {  
      var y, z;  
      ...  
      c(z);  
      ...  
    } // end of b  
    ...  
    b(x);  
    ...  
  } // end of a  
  function c(plums) {  
    ... <-----1  
  } // end of c  
  var l;  
  ...  
  a();  
  ...  
} // end of bigsub
```


Q25. Although local variables in the methods of C-based languages are allocated at the beginning of each activation, under what circumstances would the value of a local variable in a particular activation retain the value of the previous activation?