Tutorial - 02

Page No.:
Date : / /

Only non-coding questions

Q-4 Here is a structure declaration.

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

a) write a function that passes a box structure by value and that displays the value of each member.

Ans

```
void point (struct box abox)
{
    cout << abox.marker << "\n";
    cout << "Dimensions : " << abox.height << " ";
    cout << a.box.length << " ";
    cout << abox.width << "\n";
    cout << "Volume : " << aboxvolume << "\n";
```

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.

Ans

```
void calcVolume (struct box *abox)
{
    abox -> volume = abox->length * abox->width
                    * abox->height;
}
```

c) write a simple program that uses these two functions

Ans

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

int main()
{
    box box1;
    cout << "Enter length, width and height of
            box respectively: ";
    cin >> box1.length >> box1.width >> box1.height;
    calcVolume (&box1);
    print (box1);
```

d)

```
void print2 (box &abox){
    cout << abox.marker << endl;
    cout << "Dimension : " << abox.length << " ";
    cout << abox.width << " ";
    cout << abox.height << endl;
    cout << "Volume : " << abox.volume << endl;
```

e) Write a function that has a reference to a box structure as it's formal argument and sets the volume member to the product

=> void calcVolume2 (box &abox)
{
abox.volume = abox.length * abox.height * abox.width;
}

Q6 Suppose the song() function has this prototype

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

a) How would you modify the prototype so what the default value for times is 1

Ans void song (const Char* name, int times = 1);

b) What changes would you make in the function definition?

Ans void song (const char* name, int times = 1) {....}

c) Can you provide a default value of "O, my papa" for the name?

Ans Only if times also has a default value:

void song (const char* name = "O, my papa", int times = 1);

Page No.:
Date: / /

Q-7 Which of the following Initializations are legal? Explain why.

a) int i=-1, &r=0;

Ans Illegal as reference variable r cannot be initialized by a constant.

b) int *const p2 = &i2;

Ans legal.

c) const int i=-1, &r=0;

Ans Illegal as same reason as (a).

d) const int *count p3 = &i2;

Ans legal

e) const int* p1 = &i2;

Ans legal

f) const int &const r2;

Ans Illegal: const reference variable requires to be initialized.

g) const int i2=1, &r=i;

Ans legal

Page No.:
Date : / /

Q-8 Explain the following definitions. Identify any illegal.

a) int i, * const cp;

Ans Illegal as int * const cp requires initialization

b) int *p1, *const p2;

Ans same as (a)

c) const int ic, *cp = ⁣

Ans Illegal

d) const int* const p3;

Ans Illegal

e) const int* p;

Ans legal

f)

Q-9 Using variable in the previous exercise, which of the following assignments are legal? Explain why

a) p = ic;

→ legal → const int can be used to initialize int

b) p1 = p3;

illegal → const* int* cannot initialize int*.

c) p1 = ⁣

Illegal → const int* cannot initialize int*

d) p3 = ⁣
→ Illegal → const int* const cannot be assigned new value

e) p2 = &p1;
→ Illegal → int *const cannot be assigned new value

f) ic = *p3
→ Illegal → cannot assign const int a new value

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];
→ Legal ~~int siz~~ unsigned can be typecasted to size_t

b) int ia[4*7-14];
→ Legal → int can be typecasted to size_t.

c) int ia[txt_size()];
→ legal → function returns int type cast to size_t

d) char st[11] = "fundamental";
→ Illegal → fundamental is char[12] thus cannot initialize char[11].

Page No.:
Date : / /

Q11 Correct errors in each of the following code fragments

a)
```
if (ival1 != ival2)
        ival1 = ival2;
else ival1 = ival2 = 0;
```

b)
```
if (ival1 < minval)
{   minval = ival;
    occurs = 1; }
```

c)
```
if (int ival = get_value())
    cout << "ival = " << ival << endl;
else
    cout << "ival = 0\n";
```

d)
```
if (ival == 0)
  ival = get_value();
```

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);
```

-s Ans -> Loop print the sum of 2 integers as long as valid input is provided.

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

Ans Repeat previous task if get_response() returns non-zero value

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

Ans Same as (b), but ival is checked in loop.

Q-(2) 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;
}

Ans error :- return type does not match with returned argument.

b) f2(int i) { /* - - - - */ }

Ans error : no return type specified, use
void f2(int i) { /* - - - - */ }

c) int calc (int v1, int v1) { /* - - - - */ }

Ans error : v1 already declared, we can use different variable name like v2 and no return statements

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

Ans error: need to write funtion body in parenthesis.

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.

Any

```
class X {
        Y* y;
};
class Y {
    X x;
};
```

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

a) A class must provide atleast one constructor

Any False → If no constructor provided, cpp uses defaut constructor.

b) A parameter list is empty in default constructor.

Ans True

c) If there are no meaningful default values for a class, the class should not provide a default constructor.

Ans True

d) If a class doesn't define a defaut constructor, the compiler generates one that initializes each data member to default value of it's associated type

Ans False → ~~compiler~~ only generates default constructor

Page No.:
Date: / /

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 enter key at the end of each line:

Hi!
Send $10 or $20 now!

What is the output

Hi!$
Send$ $ ct1 = 9, ct2 = 9

Page No.:
Date : / /

Q19 Consider the following skeletal C program :

```
void fun1(void);
void fun2(void);
void fun3(void);
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 variable, the name of function it was defined in.

a) main → fun1 → fun2 → fun3

fun3 → d, e, f
fun2 → c
fun1 → b
main → a

Page No.:
Date : / /

b) main → fun1 → fun3

fun3 → d, e, f
fun1 → b, c
main → a

c) main → fun2 → fun3 → fun1

fun1 → b, c, d
fun3 → e, f
main → a

d) main ~~calls~~ → fun3 → fun1

fun1 → b, c, d
fun3 → e, f
main → a

e) main → fun1 → fun3 → fun2

fun2 → c, d, e
fun3 → d
fun1 → b
main → a

f) main → fun3 → fun2 → fun1

fun1 → b, c, d
fun2 → e
fun3 → f
main → a

Page No.:
Date : / /

Q.20 Assume the following JS 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
sub2();
```

Ans for static scoping
value of x in sub1 = 5

for dynamic scoping
value of x in sub1 = 10

Q.21 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?

Ans for function base::print()
↳ It prints the basename variable of the class for function derived::print().

↳ It prints the base name it gets from base class ~~is~~ and the variable i.

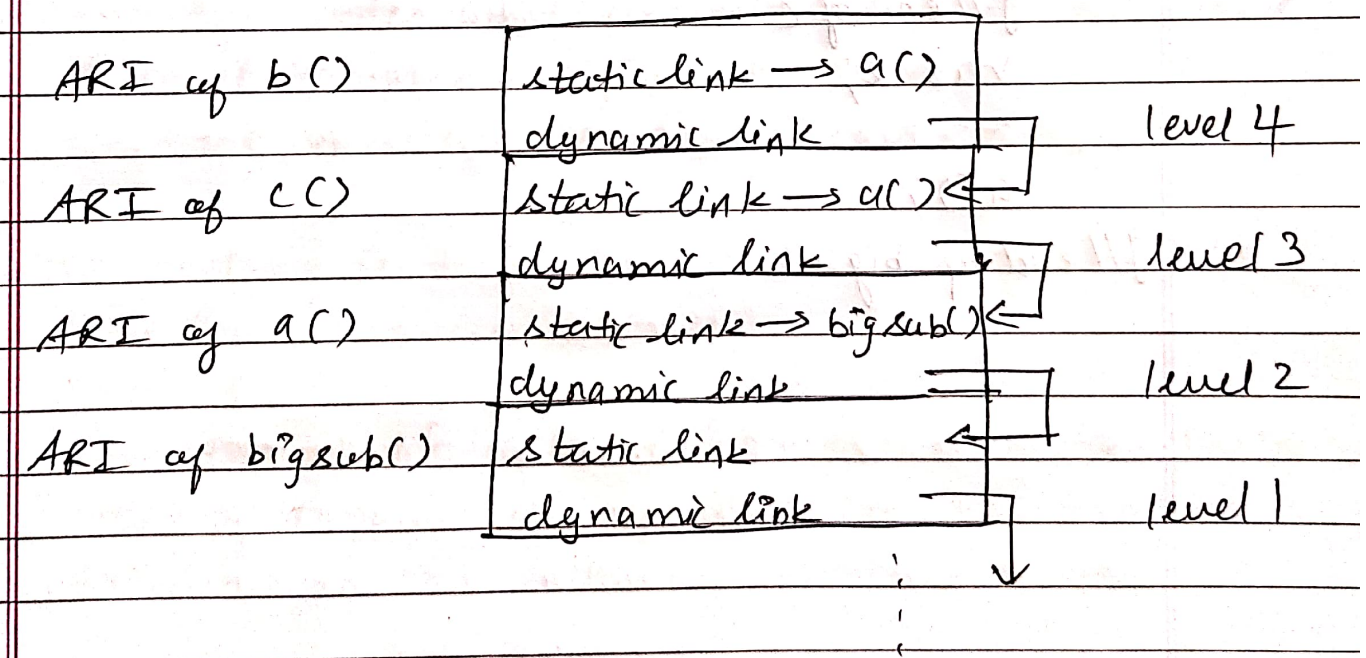
→ The problem is that it's call to print function of base class is wrong and should be
base::print(os);

Q-23 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
        ----
    } //end of a
    ----
    a();
    ----
} //end of bigsub()
```

Ans

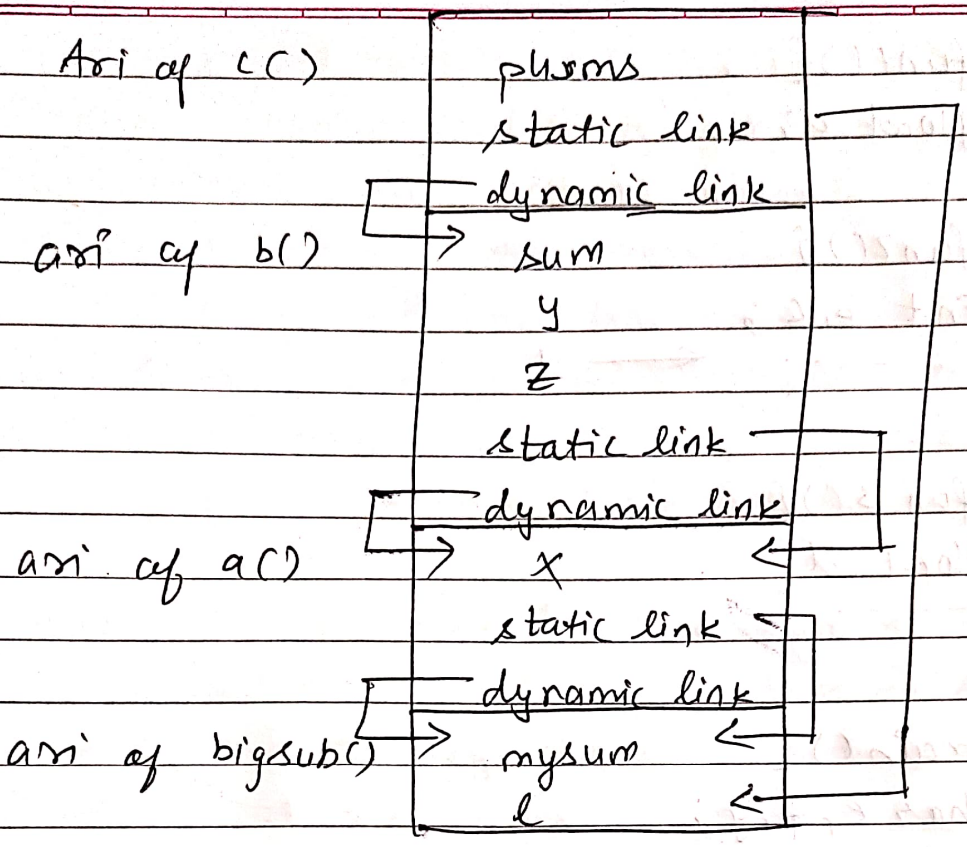


Q-24 Show the stack with all activation records instances, including static and dynamic chains, when execution reaches position 1 in the following skeletal program. Assume bigsub is at level 1

```
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){
        ---
    } // end of c
    var e;
    ---
    a();
} // end of bigsub
```

Page No.:
Date : / /

Ans



Q-25 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 local variable in a particular activation record retain the value of the previous activation?

Ans If variable is declared static it will retain it's value from previous activation.

Q-26 Show the ~~static~~ with all record activation instances including dynamic chains, when execution reaches position 1 in the following skeletal program.

This program uses deep access method to implement dynamic scoping.

```
void fun1(){
    float a;
}
void fun2(){
    int b, c;
    - - - .        <-- 1
}
void fun3(){
    float d;
    - - - .
}
void main(){
    char e, f, g;
}
```

The calling sequence for this program for execution to reach fun3 is

main -> fun1, fun1 -> fun3, fun3 -> fun2

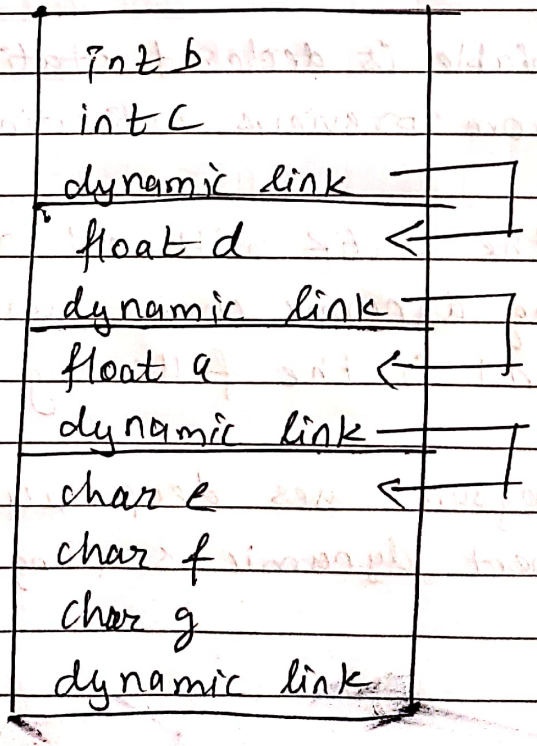
Any

AR of fun2

AR of fun3

AR of fun1

AR of main



Page No.:
Date: / /

Q-27 It is stated that in this chapter when non-local variables are accessed in a dynamic-scoped language using the dynamic chain, variable names must be stored in the activation records with the values. If this were actually done, every non-local access would require a sequence of costly string comparisons on names. Design an alternative to these string comparisons that would be faster.

Ans make a trie and for each variable name store either address of it's own stack or store a unique number to identify the variable to be used instead of direct variable names in shallow or deep access methods respectively.

Q-28 Show the stack with all activation record instance 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(flag) {
        function b() {
            a(false);
            ...
        } // end of b
        ...
        if (flag
        b();
        else
        c();
        ...
```

```
} //end of a
function c(){
    function d(){
    ------
    } //end of d
---------
d();
} //end of c
a(true);
} //end of bigsub
```