This document is generated by $\LaTeX$ POPL-2

CS18BTECH11018

1

3.

The general method is for the caller to compute the callee's static link and to pass it as an extra but as a hidden parameter. If callee is nested inside caller then, caller passes it's own frame pointer to callee's static link and if callee is outside the nesting caller,then caller dereferences it's static link same number of times as the callee's scopes and passes it as static link of callee.

2

5 .

Stack pointer points to first/last(depends on machine and compiler) unused location on the stack and Frame pointer points to a location near the bottom of the frame. We need a stack pointer to add something new to the stack and we need frame pointer for debugging.Stack pointer is used when frame size is known at the compilation time but in variable size frames, frame pointer is useful to access objects with known displacement from the frame pointer. So both are required for a subroutine.

3

6 .

A subroutine with no local variables and nothing to save doesnot need a stack frame on RISC machine and few simple subroutines might not touch the memory at all. So they can take the arguments in registers and entire computation can be done only in registers(caller-saves).

4

8 .

Whenever we have a possibility of performing task in the caller or the callee, we prefer to do it in the callee because, in calleesaving mechanism we can save data in registers which can retrieved muchfaster than the stack.

5

9 .

When we allocate the space for arguments in the stack, even when they pass them in registers, we don't need any special method to access parameters of register, we can handle all parameters similarly, so that it makes it easy for us.

6

11 .

Both macro and in-line subroutine expand they are invocated but the way compiler handles them is different.While the inline functions are expanded during compilation , macros are expanded when the program is processed by the preprocessor.

7

19 .
When a variable is returned by reference, a reference to the variable is passed back to the caller. The caller can then use this reference to continue modifying the variable, which can be useful at times. Return by reference is also fast, which can be useful when returning structs and classes.

So consider a case where we immediately want to modify the output of a subroutine without creating a new copy and just modifying the original returned variable, then return by reference is useful.

8

22 .

A parameter which need to be passed necessarily by the caller and which takes some default value by not passed by the caller is default parameter .
The implementation is pretty easy. If any argument is missed,then the compiler generates a calling sequence that loads those defaults into registers or pushes them onto the stack, as appropriate.

9

24 .
Variable length argument lists allow function to act based on the number of arguments given.
Java and C support variable numbers of parameters but they do so in a type-safe manner, by requiring all trailing parameters to share a common type unlike C and C++.

10

27 .
Subroutines can be used when you need to pass arguments, get a return value, or activate the independent execution of logic but macros can be used when defining run-time interface parameters.

11

30 .
Constraints specifies the kind of types allowed with the generic parameters. If the user explicitly declares the operations permitted on a generic parameter they are called explicit constraints. Sometimes the language implementation enforces some constraints based on the context, on the generic parameter, these are called implicit constraints.

12

35 .
we have to produce a table containing starting address of a block of code and the address of corresponding exception handler during compile time and sort the table in the order of 1st field. In case of exception, we search for its handler in the table using binary search, with program counter as a key. So this way, we avoid additional costs in case of no exceptions as this whole search process occurs only when exception arises.

13

39 .
Setjmp uses buffer to remeber current position and returns 0 and longjmp is used to go back to the place buffer is pointing to. Longjmp routines have undefined behaviour once the program exits the protected code and setjmp has limited extent for storing information.Cache coherence problems may also

arise because of longjmp.

14

40 .
A Volatile variable is type of of variable which can change it’s value in memory spontaneously and it is useful in conditions where there is a possibility of loosing the changes that were cached in registers will be lost.

15

42 .
The major difference between coroutines and a threads is that co-routines are cooperatives whereas threads are preemptive. With coroutines user should decide when to switch coroutines.

16

46.
A discrete event simulation is one in which the model is naturally expressed in terms of events that happen at specific times. This is the most important application of coroutines.

17

47 .
An event is an action or occurrence to to which programming which is currently running needs to responds. It may be something like user input from some console, they can happen arbitrarily.

18

8.3
Program in C:
Compiler : Gcc

```
# include <stdio.h>
void foo(int x, int y, int z)
{

}
int argumentOrder(int n)
{
printf(”Argument evaluated now is %d”, n);
return n;
}
int main(void)
{
foo(argumentOrder(1), argumentOrder(2), argumentOrder(3));
return 0;
}
```

Output:

```
Argument evaluated now is 3
Argument evaluated now is 2
Argument evaluated now is 1
```

Clearly from the output arguments are evaluated from right to left as per print statements in the language and compiler that i have chosen. But this depends on compiler and other pragmatics and things might vary.

19

8.4
It could be possible that with that architecture of the machine and compiler, that variable might have been initialized to 0 and value isn't being lost when

function is recalled. But this might not be same on all machines , so behavior on other systems might be different, or nondeterministic.

20

8.8

Consider case where we have Case 1 : shift(x,10,0); here x becomes 10 and shift ends but consider case where we have
Case 2 : (x,y,0), then x value changes to that of y's. and y becomes 0.
When we pass shift(x,y+0,0) then some value like 10 as seen is returned by y+0 and x value changes to that of y's and then shift ends. But with compiler optimization, y+0 is just substituted as y and then y also becomes 0 as explained.

21

8.14

foo is a function taking a pointer to a function taking a double and a pointer to a double that returns a double and a double that returns a pointer to a function taking a double and anything that returns a double.

22

8.15

No program doesn't run faster when the programmer leaves optional parameters, infact it gets slower because , it has to load these parameters from some other package unlike getting getting parameters from registers which is much faster.

23

8.29
Whenever compiler gets to know that object is about to go out of scope, then the corresponding destructor must be invoked by compiler. The destructor then should de-allocate all the resources used by that object.

Chapter 9

24

1 .
Encapsulation, Inheritance , Polymorphism are generally considered to be the three defining characteristics of object-oriented programming.

25

3 .
Abstraction is selecting data from a larger pool to show only the relevant details to the object. It helps to reduce programming complexity and effort, reduces the load of computation.
It minimizes programmers effort,gives independence to programmer, it makes updating a program easier.

26

6.
If an entity of a class is defined as private then it can be can accessed by the entities from that class only. It is used to increase the security of program and avoid unnecessary exposure of the data.

27

7.
It is the scope resolution operator.It can be used when we are defining a member function of the class outside the class, then we can use the scope resolution operator.It is also used to access global variable when there is a local variable with same name in the present scope.

28

10.
Constructors and destructors are special member functions of the class. The Compiler calls the Constructor whenever an object is created and the destructor whenever object is about to die.

29

14.
This pointer is used to represent the address of an object inside a member function.this pointer is used to represent the address of an object inside a member function.One important application of this pointer is distinguishing data members from local variables of member functions if they have same name.

30

16.
1.Public : Any class member defined as public is accessible to everyone.
2.Private : Any class member defined as private is accessible only by the functions inside the class.
3.Protected : Any class member defined as protected is accessible to subclasses also.

31

20.
Class which is declared inside another class is called nested class.Nested classes are divided into two categories: static and non-static. Nested classes that are declared static are simply called static nested classes. Non-static nested classes are called inner classes.

32

22.
An extension method is a static method and must be located in a static class. It enables us to add methods to existing types without creating a new derived type, recompiling, or modify the original types.

33

23.
No constructor doesn't allocate the space for an object. Some other mechanism takes care of it, constructor just takes care of initialisation in allocated memory.

34

25.
If we have variables as references, we should create objects explicitly whereas, if variables are values, then object creation can occur implicitly. If objects are created explicitly by us we can have control over over the appropriate constructor to be called but we lose this flexibility in case of value model, so object initialization simpler in a language with a reference model of variables.

35

28.
When we create object of a class, its constructor is called. But when we create an object of a class that is inherited from some other class, then first the constructor of the parent class is called then the constructor of that class is called.

36

29.
Initialization is creating a variable and getting it ready for further use. Assignment is one way of initialization but not the only way.

37

31.
The binding which can be resolved at compile time is static binding but in dynamic binding compiler doesn't decide the method to be called. Binding of all the static, private and final methods is done at compile-time whereas overriding is an example of dynamic binding.

38

35.
With dynamic binding, we can have methods with same name having different implementation in different classes by deciding the actual method to be invoked at run time based on the object, thus allowing run-time polymorphism.

39

40.

Abstract class has abstract methods. Abstract methods are nothing as pure virtual methods. Abstract classes can't have instances.

40

43.

Creation of object closures rely on dynamic method binding. We allow arbitrary subroutines with arbitrary number of arguments to be scheduled using dynamic method binding.

41

9.2

42

9.14

In Java we cannot have member methods with same signature but Non-Virtual member functions of C++ allows to have same function signature in inheriting classes .Virtuality in C++ takes care of calling appropriate method.

43

9.17

b just can acts as an integer overwriting previous its type char.
That is if we have an object of bar say
bar temp;
temp.b will be an integer implicitly but temp.foo::b will be char pointer.

44

9.21

Yes we can have objects of classes derived from foo assigned to foo*, but we can't have the type foo because , abstract classes can't have instances.(objects)