Common Data for Q30 & Q31 is given below:

Consider the program given below:

```
main ( )
{
        integer x;
        x = 5;
        P(x);
        Q(x);
        print (x);
}

Q (integer z)
{
        z = z +x;
        print (z);
}
P (integer y)
{
        integer x;
        x = y+2;
        Q(x);
        print(x);
}
```

Common for 30 & 31 →

```
main()
{ int x;
  x=5;
  P(x);
  Q(x);
  printf(x);
}
```

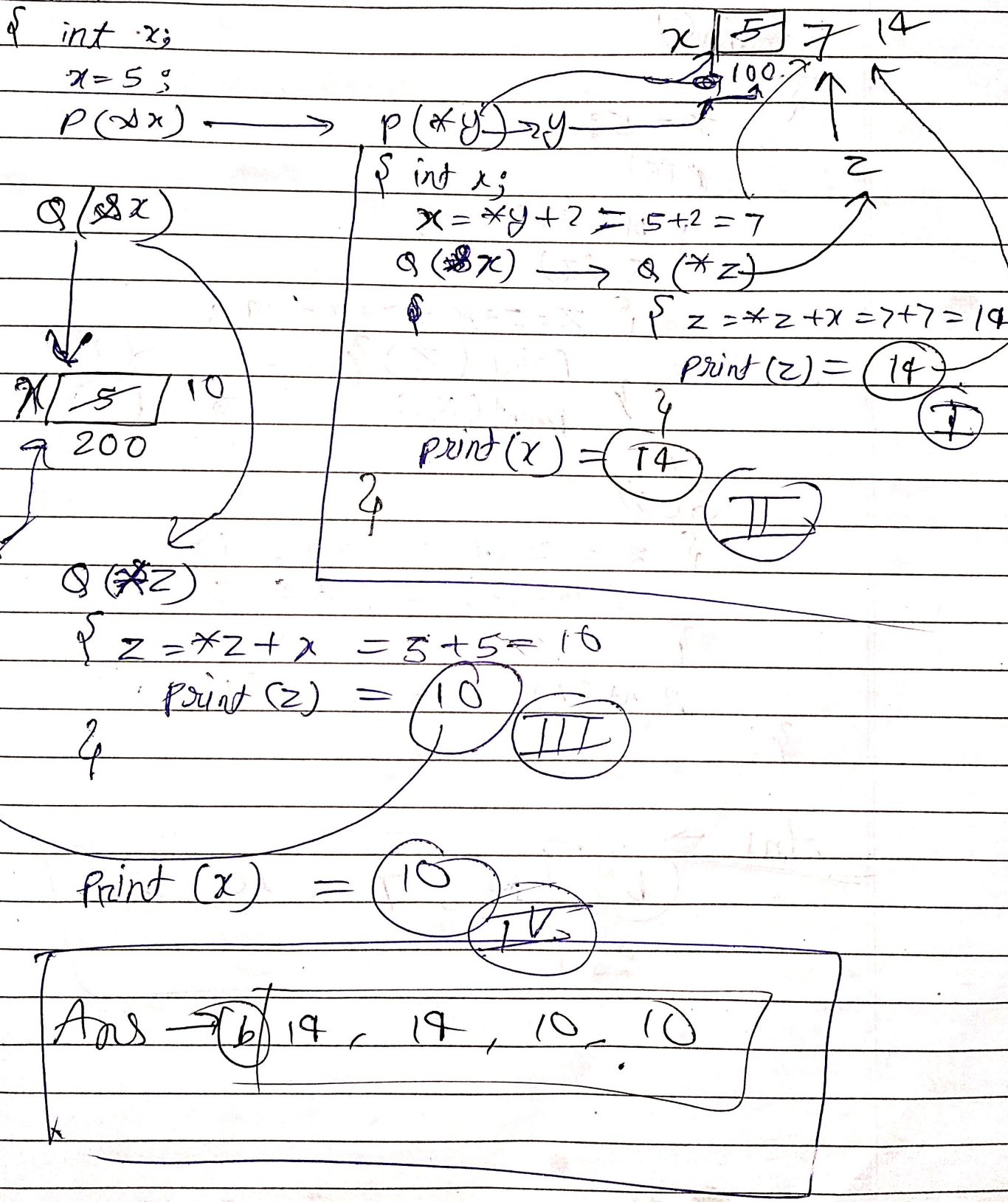
```
Q(int z)
{ z=z+x;
  printf(z);
```

```
P(int y)
{
  int x;
  x=y+2;
  Q(x);
  printF(x);
}
```

30. What is the output of the program, when the parameter passing mechanism is call by reference and the dynamic scoping is used?

(a) 12, 7, 10, 5
(b) 14, 14, 10, 10
(c) 14, 7, 10, 5
(d) 12, 12, 10, 10

Q 305 Output -? when - Call by reference
Dynamic scoping.



Ans → (b) 14, 14, 10, 10

Teacher's Signature .....

31. What is the output of the program, considering call by value and dynamic scoping?

(a) 12, 12, 10, 10
(b) 14, 7, 10, 5
(c) 14, 14, 10, 10
(d) 12, 7, 10, 5

(b) 31⇒ Output-?

Call by Value
Dynamic Scoping –

```
main()
{
    x = 5;
    P(5)
    { x = 5+2 = 7;
      Q(7)
      { z = z+x = 7+7 = 14;
        print(z) } ——————→ (14)
      } print(x); ——————→ (7)
    }
    Q(5)
    { z = z+x = 5+5 = 10
      print(z) ——————→ (10)
    }
    print(x) ——————→ (5)
}
```

Ans → (b) | 14, 7, 10, 5

32. The object – oriented paradigm includes which of the following properties?

I. Encapsulation

II. Inheritance

III. Recursion

IV. Rule – based evaluation

(a) I & II

(b) I, II, III

(c) I, II, III, IV

(d) I, II, IV

(a) 32⇒ Object - Oriented (include)

(i) Encapsulation (✓)
(ii) Inheritance (✓)
(iii) Recursion (✗)
(iv) Rule - based Evaluation (✗)

And Also → Polymorphism (✓)

Ans ⇒ (a) I & II

33. Consider the following pseudo code

```
x = 1;
y = 1; i = 1;
while (x ≤ 1000)
{
        x = 2^x;
        i = i + 1;
}
```

What is the value of 'i' at the end of the pseudo code?

(a) 4 (b) 7

(c) 5 (d) 8

(C) (33) Psedu Code :-

```
x = 1 ;
i = 1 ;
while (x <= 1000)
{  x = 2^x ;
   i = i + 1 ;
}
```

Output i = 5

Ans - (C) i = 5

①	$(1 \le 1000)$ $x = 2^1 = 2$ $i = 2$	$i = 2$
②	$(2 \le 1000)$ $x = 2^2 = 4$ $i = 3$	$i = 3$
③	$x = 2^4 = 16$	$i = 4$
④	$x = 2^{16} = 65536$	$i = 5$
~~⑤~~	~~$x = 2^{256}$ (false)~~	
⑤	$x = 2^{65536}$ (False)	

Teacher's Signature .....

34. Consider the following statements:

I. With dynamic scope complete type checking can not be performed until runtime.

II. Dynamic scope is used by languages such as C & C ++.

III. Use of dynamic type checking, instead of static type checking allows faster Program execution.

Which of them is/are true?

(a) I Only
(b) I & II
(c) II, III
(d) I, II, III

(a) 34) True / False -

True (i) With dynamic scope complete type checking can not be performed until runtime.

False (ii) Dynamic scope is used by languages such as C & C++.

May not be (iii) Use the dynamic type checking, instead of static type checking allows faster program execution.

False

ii → C support dynamic scope but C++ does not support

Ans ⇒ (a) I only → True

35. In a system with automatic memory management, a garbage collector typically has the responsibility for reclaiming allocated memory chunks whose contents cannot affect legal computation. Such objects are identified by determining that they cannot be reached from a root set, which of the following is ***NOT*** part of the root set in a typical garbage collector?

(a) Actual parameters of the active procedures.

(b) Dynamically allocated objects on the heap.

(c) Global variables of the program.

(d) Values in machine registers.

(b) 35⇒ Which is not part of the root set in a typically garbage collector?

Ans → (b) → Dynamically allocated objects on the heap.

Common Data for Q36 & Q37 is given below.

Consider the program given below with procedures P_1, P_2, P_3. The main program call procedures P_1, P_1 calls P_3, and P_3 calls P_2.

```
program main;
    i : integer;
    procedure P1( )
    {       i : integer;
            procedure P2( )
            {
                    write (i);
            }
            procedure P3( )
            {       i : integer;
                    i = 5;
                    P2 ( );
            }
            i = 7;
            P3( );
    }
    main ( )
    {       i = 3;
            P1 ( );
    }
```

Common for 36 & 37 →

```
Program main;
  i : int;
  procedure P1()
  { i : int;
    Procedure P2()
    { write(i);
    }
    Procedure P3()
    { i : int;
      i = 5;
      P2();
    }
    i = 7;
    P3();
  }
  main()
  { i = 3; P1(); }
```

36. What will be printed by the program if it uses static scoping to resolve the free variables?

(a) 7
(b) 5
(c) 6
(d) none of these

(a) 36 Output → Static Dynamic Scoping

```
main()
{
    i = 3;
    P1();
}

P1()
{ i: int;
    P2()
    { print(i) }
    P3()
    { i: int;
      i = 5; // local for P3()
      P2();
    }
    i = 7; // global for P3()
    P3();
}
```

(I) Call P1() → In Main()
(II) P1() call P3()
(III) P3() call P2()
(IV) P2() → print(i) = 7 → global (i)

i = 3
i = 7
i = 5

Ans → (a) (7)

37. What if in program of question 36, uses dynamic scoping?

(a) 7
(b) 5
(c) 6
(d) none of these

(b) 37) Output-? Dynamic Scope

```
main()
{ i=3;
  p1();
}
```

```
p1()
{ i: int;
  p2()
  { print(i)}
  p3()
  { i; int;
    i = 5; // loco
    p2();
  }
  i=7;
  p3();
}
```

(i) P1() call ⇒ In Main()
(ii) P1() calls P3()
(iii) P3() calls P2()
(iv) P2() print(i) = 5

i = 3
i = ~~7~~
i = 5

↳ because i=5 is dynamic and visible to P2() fn.

Ans → (b) (5)

38. Structures in C/C++ Support:

I. Call – by – Value

II. Call – by – Reference

III. Call – by – Name

IV. Call – by – Text

(a) I, II, III

(b) II, III

(c) II only

(d) I & II

38 ⇒ C/C++ Support →

(I) Call by Value
(II) Call by reference
(III) Call by name
(IV) Call by text

Ans → (a) I, II, III

Note: → C++ does not support call by text.