Lecture 1 CTQ 12/08/2020

Programming and Data Structures

12 - 16 marks

1. Programming → parameter passing techniques
   - 2 marks → Scope of a variable
     - → static scoping
     - → dynamic scoping
     - 3-4 examples
   - 2 marks → static variable — 5-6 examples
   - 4 marks → pointers — 25-30 examples
   - 2 marks → structures, files — 3-4 examples

   min. 5 marks — max. 10m

2. Arrays — 1D, 2D, 3D — 2-4 marks
3. Linked List — 2-4 marks
   - SLL (tricky)
   - C-SLL
   - DLL (easy)
   - C-DLL
4. Trees — 3-5 marks
   - BT
   - BST
   - AVL
5. Stack — 2 marks
   - recursion
   - TOH
   - infix to postfix
   - postfix evaluation
6. Queue — 2 marks
   - LQ
   - CQ
   - DEQ
   - PQ

Scope of a variable

1) Static scoping (most modern languages)
2) dynamic scoping (old languages)

* most programming languages prefer using static scoping

Q1 For the following code print the output if
a) static scoping
b) dynamic scoping are used

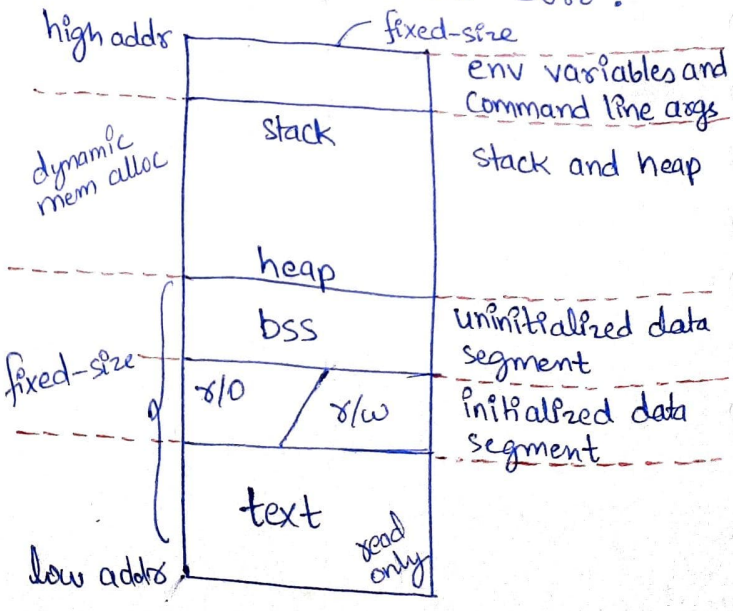
```
var a, b : integer ;
procedure p;
    a := 5;
    b := 10;
end(p);
Procedure Q;
    var a, b: integer;
    p;
end(Q);
begin
    a := 1;
    b := 2;
    Q;
    write(a,b);
end
```

mem allocated in stack during run-time

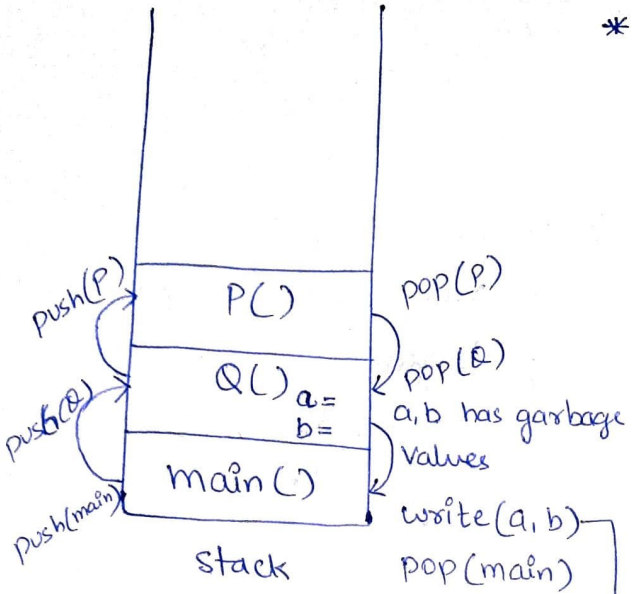
main

* declaration - creating memory
* memory will be allocated to global variables in compile time
* if mem is allocated to any variable in stack area, the default value will be garbage
* if mem is allocated to any variable is static area, the default value will be zero.

* global and static variables are stored in data segment
* auto variables are stored in stack
* register vars in CPU registers



* whenever program is running stack is used

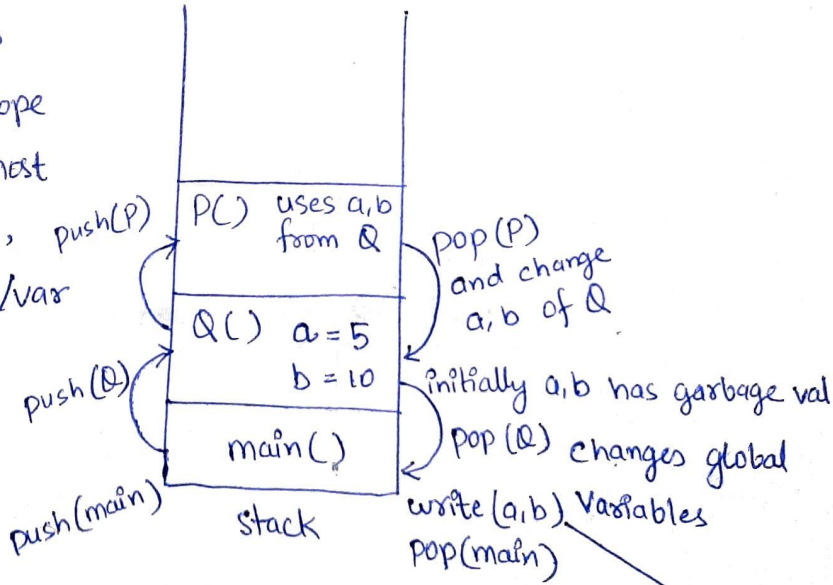


Static mem: a = Ø1 5, b = Ø2 10

Output: 5, 10

* Static Scoping
  * When main is called, a and b are used without declaring them, ofcourse inside main().
  * variables a and b are called free variables as we don't know where they came from.
  * Static scoping means when you don't have the local memory, use the global memory
  * in other words, static scoping means if the variables are not found in local scope, look for them in the global scope.
  * if scope is not mentioned, use Static scoping

* Dynamic Scoping
  * it says if local mem is not present then use the most recent mem
  * in other words, if variables are not found in the local scope then look for them in the most recently used scope, if found, Use them else use global mem/var at last



Static mem: a = Ø1, b = Ø2

Output: 1, 2

* Different outputs according to the type of scoping used.

Q2 Output of program using (i) static scoping (ii) dynamic scoping

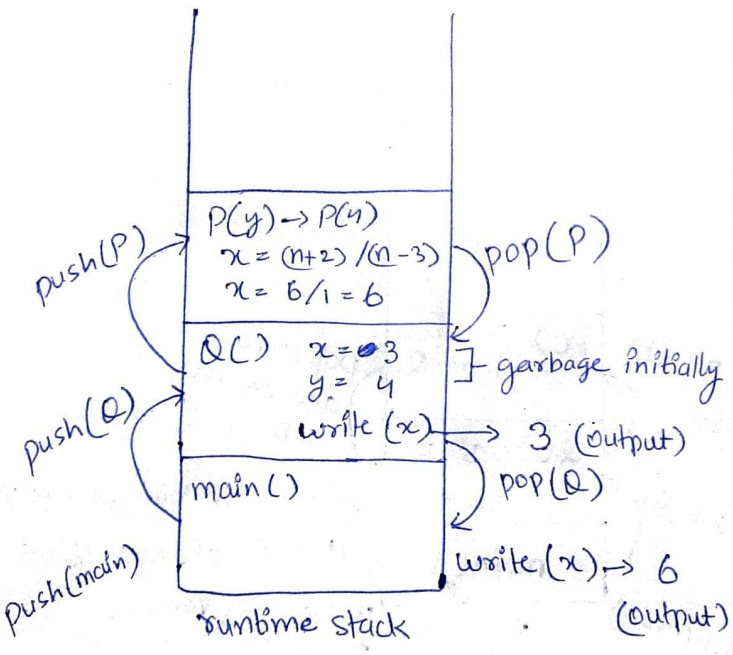
```
var x, y : integer ;
procedure P (n : integer) ;
    begin
        x := (n+2) / (n-3) ;
    end
begin
    x := 7 ;
    y := 8 ;
    Q ;
    write (x) ;
end

procedure Q
    var x, y : integer ;
    begin
        x := 3 ;
        y := 4 ;
        P(y) ;
        write (x) ;
    end
```

P

main

Q

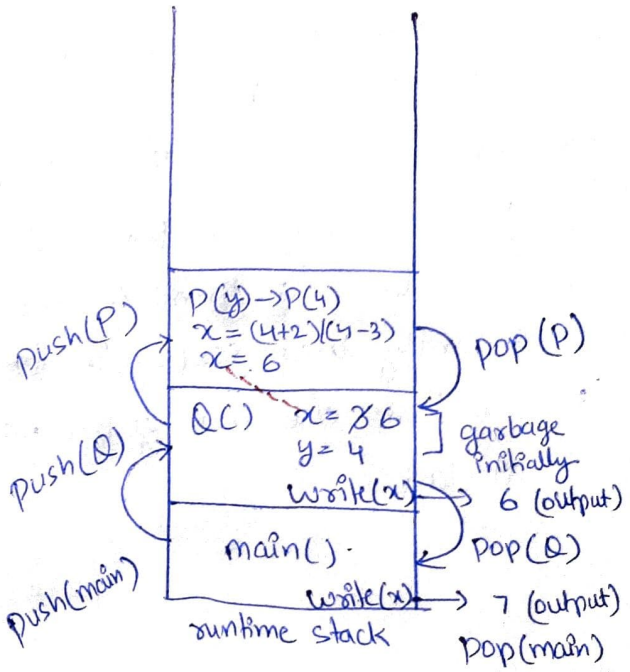
Static Scoping



static mem: x = 6, y = 8

Output: 36

Dynamic Scoping



static mem: x = 7, y = 8

Output: 67

Q3 What will be the output using (i) static scoping (ii) dynamic scoping for the following program

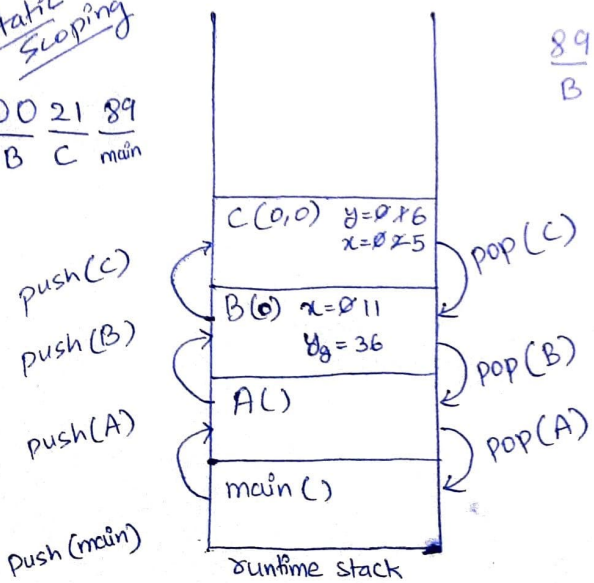
```
int x=5, y;
main() {
    x = 3;
    A();
    printf(x, y);
    B(x);
}

A() {
    printf(y);
    x = 7;
    B(y);
    x = 8;
    y = 9;
}

B(int x) {
    printf(x, y);
    C(x, y);
    x = 11;
    y = 36;
}

C(int y, int x) {
    x = 2;
    y = 1;
    printf(x, y);
    x = 5;
    y = 6;
}
```

	Static scoping	dynamic scoping
Output	0 0,0 2,1 8,9 8,9 2,1	0 0,0 2,1 8,9 8,9 2,1

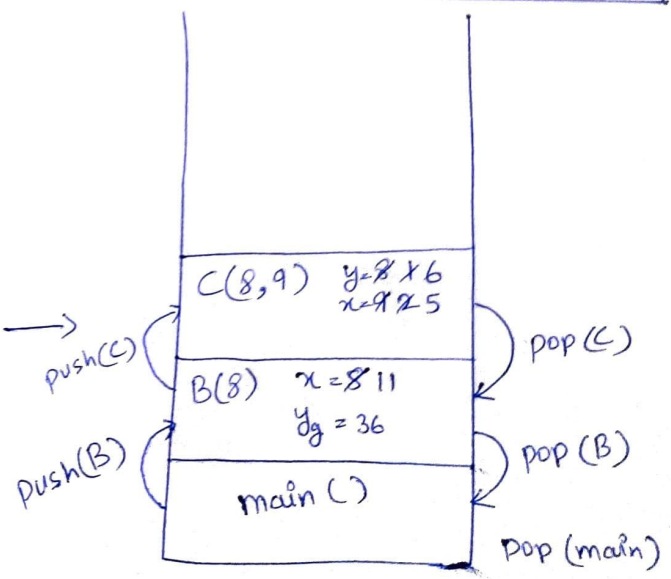
Static Scoping

0 0 0 2 1 8 9
A B C main

8 9 2 1
B C



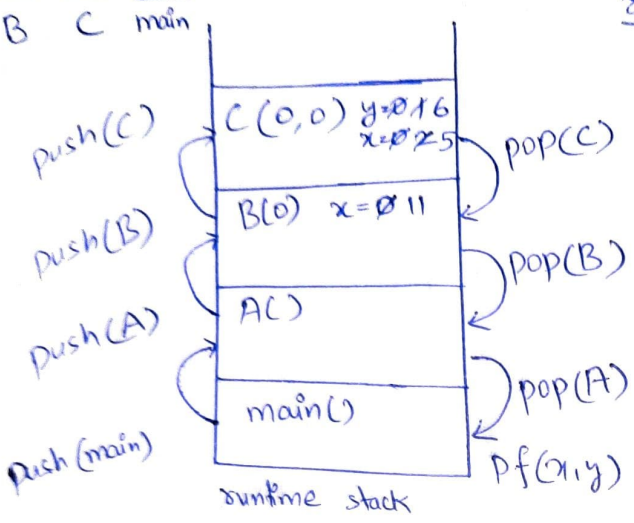
Static mem: x = 8 11 8
y = 0 36 9



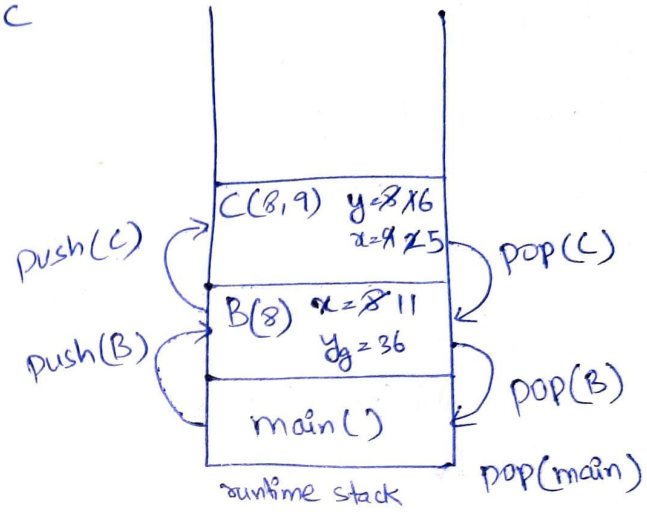
Static mem: x = 8
y = 9 36

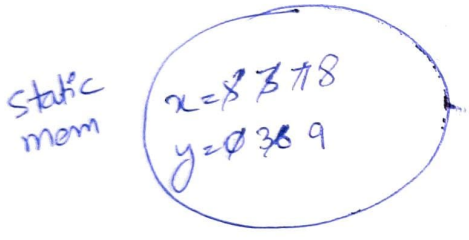
Dynamic Scoping

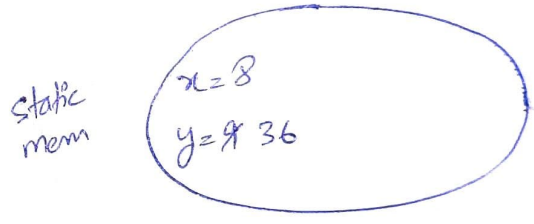
0 00 21 89
A B C main



89 21
B C







* more push/pop operations are performed (overhead in implementation) in dynamic scoping as compared to static scoping hence static scoping is preferred.

Lecture - 2 CTQ PDS

Static Variable

Q1 The value of j at the end of the following C- program

```
main() {
    int i, j;
    for (i=0; i<=4; i++)
        j = incr(i);
}
```

```
int incr(int i) {
    static int count = 0;
    count = count + i;
    return count;
}
```

runs one time → will get stored in static mem.

my soln

i=0 0<=4
j = incr(0) count = 0 / 3
C = 0+0
C = 0
j = 0

i=1 1<=4
j = incr(1)
C = 0+1
C = 1
j = 1

i=2 2<=4
j = incr(2)
C = 1+2
C = 3
j = 3

i=3 3<=4
j = incr(3)
C = 3+3
C = 6
j = 6

i=4 4<=4
j = incr(4)
C = 6+4
C = 10
j = 10

* for static var mem will be allocated only once in static area
* only once initialization
* default zero
* output w/o static = 4
* output w static = 10

} value of j

Q2 The output of executing the following C-program is

```
int total (int V) {
    static int count = 0;
    while (V) {
        count += V & 1;
        V >>= 1;
    }
    return count;
}
```

```
void main () {
    static int x = 0;
    int i=5;
    for (; i>0; i--) {
        x = x + total (i);
    }
    printf ("%d", x);
}
```

↓

23

my Soln x = 0 2 5 10 16 23

count = 0 1 2 3 4 5 6 7

i=5 i>0

x = 0 + total(5) = 2

→ w(5) ✓

c = 0+5&1 = 1

5 0101

1 0001

0001

→ C=1

V = 5 >>1

V = 0101 >>1 = 0010

V = 0010 = 2

→ w(2) ✓

c = 1+2&1 = 1+0 = 1

2 0010

1 0001

0000

→ C=1

V = 2 >>1

= 0010 >>1

V = 0001 = 1

→ w(1) ✓

c = 1+1&1 = 2

C = 2

V = 1>>1

V = 0

→ w(0) X

i=4 i>0

x = 2 + total(4) = 5

→ w(4) ✓

c = 2+4&1 = 2

4 0100

1 0001

0000

C=2

V = 4>>1

= 0100>>1

V = 0010 = 2

→ w(2) ✓

C = 2+2&1 = 2

V = 2>>1

V = 1

→ w(1) ✓

C = 2+1&1 = 3

V = 0

→ w(0) X

i=1 i>0

x = 16 + total(1) = 23

→ w(1) ✓

C = 6+1&1 = 7

V = 0

→ w(0) X

i=3 i>0

x = 5 + total(3) = 10

→ w(3) ✓

C = 3+3&1

C = 4

V = 0011>>1

V = 1

→ w(1) ✓

C = 4+1&1

C = 5

V = 1>>1 = 0

→ w(0) X

i=2 i>0

x = 10 + total(2) = 16

→ w(2) ✓

C = 5 + 2&1 = 5

V = 1

→ w(1) ✓

C = 5+1&1 = 6

V = 0

→ w(0) X

Q3 Consider the C-function given below

```
int f (int j) {
    static int i = 50;
    int k;
    if (i == j) {
        printf ("Something");
        k = f(i);
        return 0;
    }
    else
        return 0;
}
```

error - Stack Overflow

which one of the following is true?

x a) The function return 0 for all values of j

x b) The function print the string "something" for all values of j

x c) The function returns 0 when j = 50

(my ans) ✓ d) The function will exhaust the runtime stack or run into an infinite loop when j = 50

Q4 What will be the output of the following C-program?

```
main() {
    count(3);
}
void count (int n) {
    static int d = 1;
    printf ("%d", n);
    printf ("%d", d);
    d++;
    if (n>1) count (n-1);
    printf ("%d", d);
}
```

d = 1 2 3 4

count(3) → 3,1

↓

3>1 ✓

↓

4 ← P(d) count(2) → 2,2

↓

2>1 ✓

↓

4 ← P(d) count(1) → 1,3

↓

P(d) → 4

Output → 3 1 2 2 1 3 4 4 ✓

250

Q5 Consider the following C-code segment

```
int a, b, c = 0;
main() {
    static int a = 1;
    f();
    a += 1;
    f();
    printf("%d %d", a, b);
}
```

```
f() {
    static int a = 2;
    int b = 1;
    a += ++b;
    printf("%d %d", a, b);
}
```

my soln

main

a = 1 x 2

f()

b = 1 x 2

a = 2 + 2 = 4

pf() → 4, 2

a = 1 + 1 = 2

f()

b = 1 x 2

a = 4 + 2 = 6

pf() → 6, 2

pf() → 2, 0

f()

a = 2 x 6

Output

426200

a) 3, 1
4, 1
4, 2

b) 4, 2
6, 1
6, 1

✓c) 4, 2
6, 2
2, 0

d) 3, 1
5, 2
5, 2

13/08/2020

Q6 Consider the following C-code segment

```
int a, b, c = 0;
main() {
    static int a = 1;        /*line 1*/
    f();
    a += 1;
    f();
    printf("%d %d", a, b);
}
f() {
    static int a = 2;        /*line 2*/
    int b = 1;
    a += ++b;
    printf("%d %d", a, b);
}
```

a) 3, 1
   4, 1
   4, 2

b) 4, 2
   6, 1
   6, 1

c) 4, 2
   6, 2
   2, 0

✓ d) 4, 2
   4, 2
   2, 0

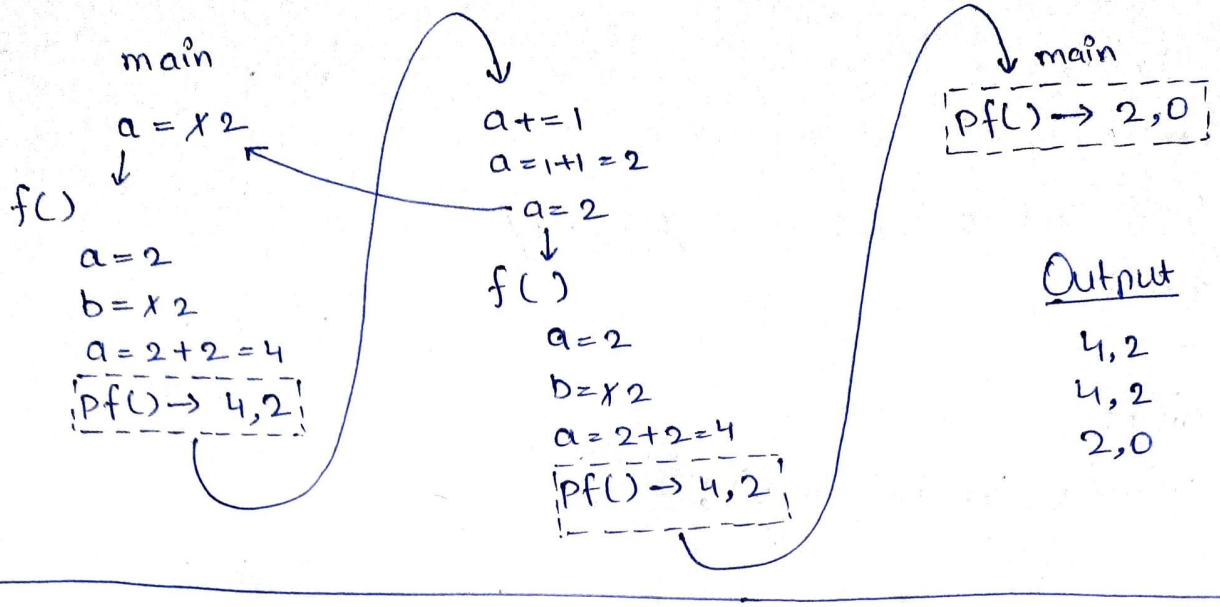
What will be the output if you replace line-1 by auto int a=1; and line-2 by register int a = 2;

Soln modified code

```
int a, b, c = 0;
main() {
    static int a=1;
    auto int a=1;
    f();
    a += 1;
    f();
    printf("%d %d", a, b);
}
f() {
    static int a=2;
    register int a=2;
    int b=1;
    a += ++b;
    printf("%d %d", a, b);
}
```

252

My soln global a=0 b=0 c=0



Pointers

Q1 The output of the following C-program is

```
int main() {
  int a=4, b=5, c=6;
  f1(a,b);
  f2(&b,&c);
  printf("%d", c-a-b);
}
```

```
void f1(int a, int b){
  int c;
  c = a;
  a = b;
  b = c;
}
```

```
void f2(int *a, int *b){
  int c;
  c = *a;
  *a = *b;
  *b = c;
}
```

my soln

a = 4
b = ~~6~~
c = ~~5~~

pf() → c-a-b
= 5-4-6
= 1-6
= -5

$f_1(4,5)$
a = 4
b = 5
c = 4
a = 5
b = 4

call by value

$f_2(\&b, \&c)$
*a = &b
*b = &c
c = *a = *(&b) = b = 5
*a = *b
⇓
b = 6
*b = c
⇓
c = 5

call by ref.

Q2 What will be the output?

```
main() {
    f();
    f();
}

swap(int *x, int *y){
    static int *temp;
    temp = x;
    x = y;
    y = temp;
}

f() {
    static int i, a = -3, b = -6;
    i = 0;
    while (i <= 4) {
        if ((i++)%2 == 1) continue;
        a = a+i;
        b = b+i;
    }
    swap(&a, &b);
    printf("%d %d", a, b);
}
```

a) a = 0, b = 3
   a = 0, b = 3

b) a = 3, b = 0
   a = 12, b = 9

c) a = 3, b = 6
   a = 3, b = 6

✓ d) a = 6, b = 3
   a = 15, b = 12

main
f() ⟶ i = 0 X 2 3 X 5
a = -3 -2 X 6
b = -6 -5 -2 3

i = 3 3<=4 ✓
(3++)%2 == 1
⇓
3%2 == 1
1 == 1 ✓
continue

i = 4 4<=4 ✓
(4++)%2 == 1
0 == 1 X
a = 1+5 = 6
b = -2+5 = 3

i = 0 0<=4 ✓
(0++)%2 == 1
⇓
0%2 == 1
0 == 1 X
a = -3+1
a = -2
b = -6+1
b = -5

i = 1 1<=4 ✓
(1++)%2 == 1
⇓
1%2 == 1
1 == 1 ✓
continue

i = 2 2<=4 ✓
(2++)%2 == 1
⇓
2%2 == 1
0 == 1 X
a = -2+3
a = 1
b = -5+3
b = -2

254

Swap (&a, &b)

*x = &a *y = &b

x	y
2000	3000
4000	5000

temp = x

x = y

y = temp

(static)

temp = ∅ 2000

x = ~~2000~~ 3000

y = ~~3000~~ 2000

1000 i = 5

2000 a = 6

3000 b = 3

⇓

Pf() → 6, 3 opt. d

main

f() ✓

f() →

i = ~~5 0 1 2 3 4~~ 5

a = ~~6 7 10~~ 15

b = ~~3 4 7~~ 12

i = 0 0<=4 ✓

(0++)%2 == 1

0 == 1 ×

a = 6 + ~~0~~ = ~~7~~

b = 3 + ~~0~~ = 4

i = 1 1<=4 ✓

(1++)%2 == 1

1 == 1 ✓

continue

i = 2 2<=4 ✓

(2++)%2 == 1

0 == 1 ×

a = 7 + 3 = 10

b = 4 + 3 = 7

i = 3 3<=4 ✓

(3++)%2 == 1

1 == 1 ✓

continue

i = 4 4<=4 ✓

(4++)%2 == 1

0 == 1 ×

a = 10 + 5 = 15

b = 7 + 5 = 12

⇓

Swap (&a, &b)

no effect on static a, b

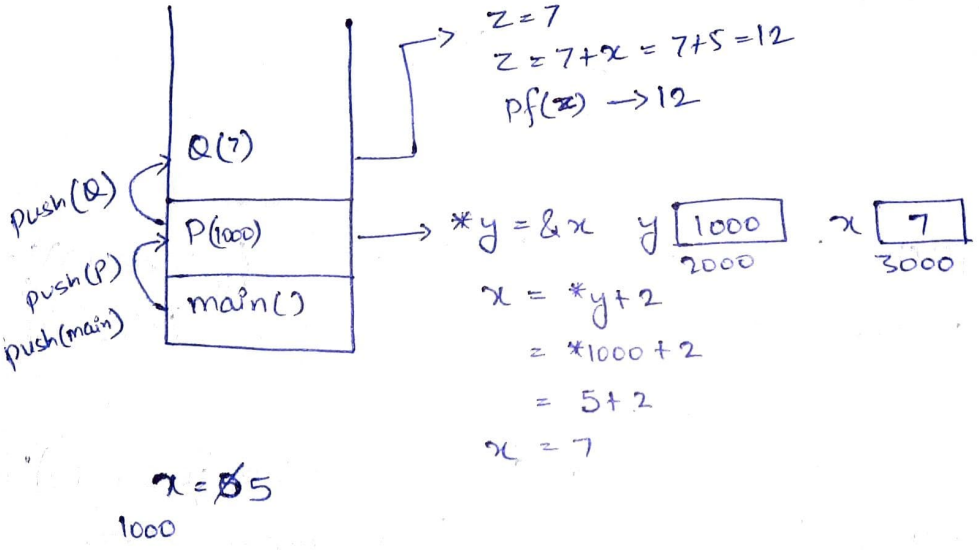
⇓

Pf() → 15, 12

Q3 Consider the following C code

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

P(int *y) {
    int x = *y+2;
    Q(x);
    *y = x-1;
    printf(x);
}

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

my soln



x = 5
1000

a) 12 7 6 ✓ b) 22 12 11 c) 14 6 6 d) 7 6 6

Lecture-4 CTQ PDS 13/08/2020

Q Choose the correct option to fill the X and Y so that the program prints an input string in reverse order. Assume that the input string is terminated by a newline char.

```
int main() {
    printf("Enter text");
    printf("\n");
    f();
    printf("\n");
    return 0;
}
```

```
void f() {
    int c;
    if (X) {
        f();
    }
    Y;
}
```

a) X is getchar() != '\n'
   Y is getchar(c);

b) X is (c = getchar()) != '\n'
   Y is getchar(c);

c) X is c != '\n'
   Y is putchar(c);

✓ d) X is (c = getchar()) != '\n'
   Y is putchar(c);

c = getchar() - read char from std input

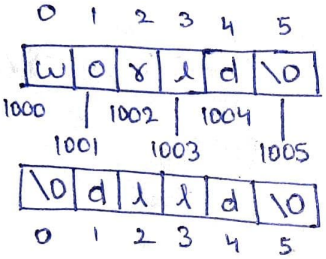
putchar(c) - print to std out char by char

Q2 What will be the output printed by the following C-program

```
int main() {
    char a[6] = "world";
    int i, j;
    for (i=0, j=5; i<j; a[i++] = a[j--]);
    printf("%s", a);
}
```

a) dlrow ✓ b) Null String c) dlworld d) worrow

My sol



i = 0 1 2 3

j = 5 4 3 2

0 < 5 ✓

a[0++] = a[5--]

⇓

a[0] = a[5]

1 < 4 ✓

a[1++] = a[4--]

⇓

a[1] = a[4]

2 < 3 ✓

a[2++] = a[3--]

⇓

a[2] = a[3]

pf() → not output (Null String)

Q3 What will be output of the following C-program segment ?

```
char inchr = 'A';        // ASCII - A => 65
switch(inchr) {
    case 'A': printf("choice A\n");
    case 'B':
    case 'C': printf("choice B");
    case 'D':
    case 'E':
    default: printf("No choice");
```

a) No choice

b) choice A

c) ✓ choice A
choice B No choice

d) No output because error is there

Q4 Consider the following C-program segment

```
void foo (char *a) {
    if (*a && *a != ' ') {
        foo (a+1);
        putchar (*a);
    }
}
```

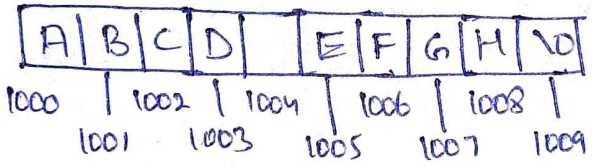
precedence
`!=`
`&&`

The output of the above code when input is "ABCD EFGH" is

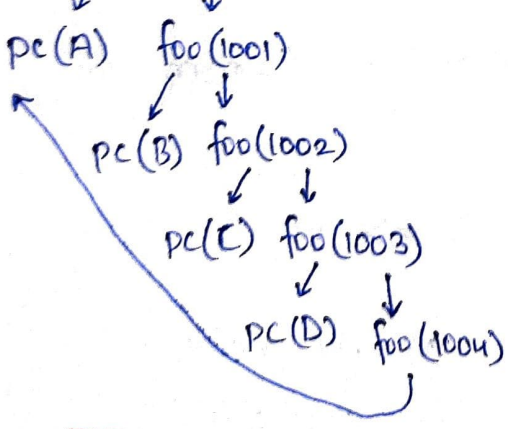
a) ABCDEFGH
b) ABCD
c) HGFEDCBA
d) DCBA

Soln

a



foo(1000)
*a && *a != ' '
(A && A != ' ') true



Output → DCBA

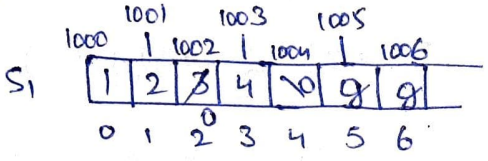
Q5

```
int main() {
    char s1[7] = "1234", *p;
    p = s1 + 2;
    *p = '0';
    printf("%s", s1);
}
```

what will be printed by the program?

a) 12 b) 120400 ✓ c) 1204 d) 1034

my sol^n



s1 [1000] (2000) p [1002] (3000) *p = '0' *1002 = '0'

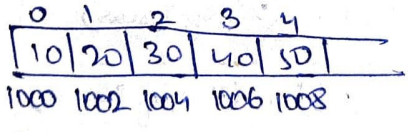
output → 1204

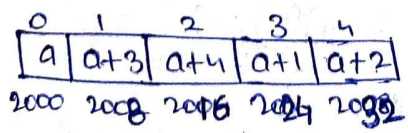
Q6

```
main() {
    static int a[] = {10, 20, 30, 40, 50};      → array of data
    static int *p[] = {a, a+3, a+4, a+1, a+2};
    int **ptr = p;                               → array of pointers
    ptr++;
    printf("%d %d", ptr-p, **ptr);
}
```

The output of the above program is 1 40

my sol





assumption
pointer size = 8B
(Same for all data types)

ptr [2000] (3000) → ptr++
= ptr + 1×8
= 2008 → ptr [2008] (3000)

ptr - p = (2008 - 2000)/8
= 1

**ptr = *(*2008)
= *(a+3) = 40

260

Q7 What is the output of the following C-code? Assume that the address of x is 2000 (in decimal) and an integer requires four bytes of memory.

```
int main() {
    unsigned int x[4][3] = { {1, 2, 3},
                             {4, 5, 6},
                             {7, 8, 9},
                             {10, 11, 12} };
    printf ("%u %u %u", x+3, *(x+3), *(x+2)+3 );
```

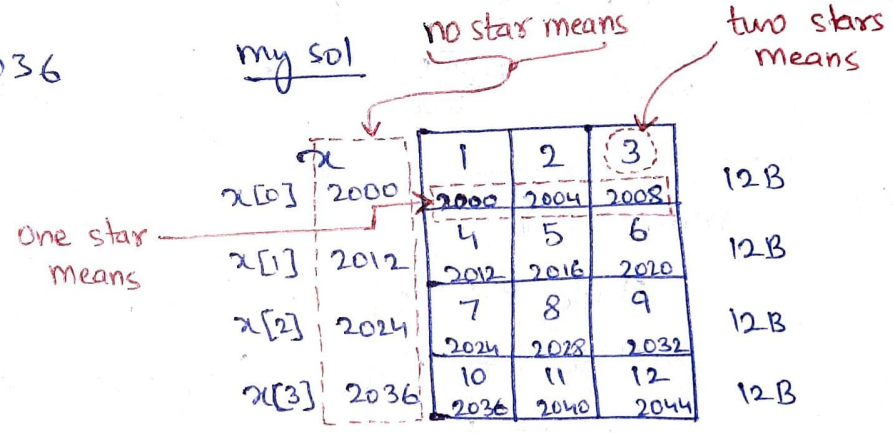
a) 2036, 2036, 2036

b) 2012, 4, 2204

c) 2036, 10, 10

d) 2012, 4, 6

my sol

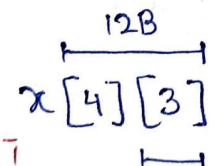


a	*a	**a
2000	2000	1

x + 3 = 2000 + 3x12 = 2036

*(x+3) ≡ x[3] = 2036

*(x+2)+3 ≡ x[2]+3 = 2024 + 3x4 = 2036



Coditions	size of data to multiply with
(i) if x+i (skipping i rows)	12B
(ii) x[m]+i (skipping i elems)	4B
(iii) x[m][n]+i	None
(iv) &x+i (skip whole array)	sizeof(x)

*x + 2
2000 + 2x4
2008

in (i) x returns address

(ii) x[m] returns address

(iii) x[m][n] returns value stored in array x

(iv) &x+i = BA of x + i x sizeof(x) → # of elems; → size of one element of int data type

here, &x+1 = 2000 + 1x4x3x4 = 2048 (skipped whole array)

Q8 Which combination of the integer variables x, y and z makes the variable a get the value of 4 in the following expression ?

$$a = (x > y) ? \{(x > z) ? x : z\} : \{(y > z) ? y : z\}$$

a) x = 3, y = 4, z = 2

b) x = 6, y = 5, z = 3

c) x = 6, y = 3, z = 5

d) x = 5, y = 4, z = 5

?: Assoc. RTL

my sol either options a or d is correct because other options do not contain number 4 in them

4(a) x = 3 y = 4 z = 2 conditional operator is RTL assoc.

a = (3 > 4) ? {(3 > 2) ? 3 : 2} : {(4 > 2) ? (4) : 2}

X skipped ✓

a ? b : c if a is true then b otherwise c

Lecture-5 PDS CTQ 13/08/2020

TRICKY QUESTION

Q1 Which one of the following choice will be printed when the program is executed?

GATE 2006
2 marks

```
struct test {
    int i;
    char *c;
}
```

*size of data types vary from compiler to compiler

2B (assumptions)

8B (sizeof pointer is same for all data types)

sizeof struct test = 10B

```
struct test st[] = { 5, "become", 4, "better", 6, "jungle",
                     8, "ancestor", 7, "brother" };
```

struct test → data type; st[] → array of struct test type

```
main() {
I    struct test *p = st;
II   p+=1;
III  ++p->c;
IV   printf("%s", p++->c);
V    printf("%c", *++p->c);
VI   printf("%d", p[0].i);
VII  printf("%s\n", p->c);
}
```

a) jungle, n, 8, ncestor

b) etter, u, 6, ungle

c) cetter, k, 6, jungle

d) etter, u, 8, ncestor

Soln

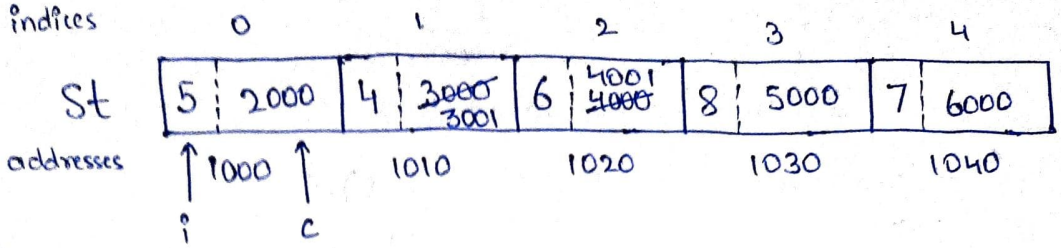
* struct size = sum of size of all data members

sizeof(test) = 10B (2B int + 8B char*)

↳ user defined data type

* data members inside struct can't be initialised as we are creating a user defined data type and not variables

* purpose of declaration is creating ~~array~~ memory

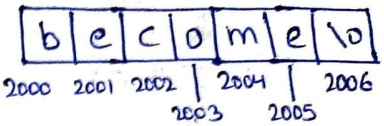


struct test type pointer

(2B) (8B)

(I) *p = st

p | 1000 | 7000



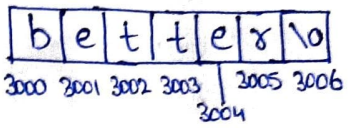
(II) p += 1

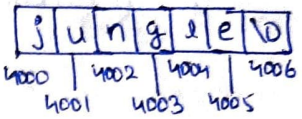
p = p+1

= 1000 + 1x10

p = 1010

p | 1010 | 7000



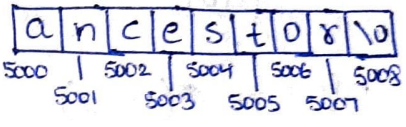


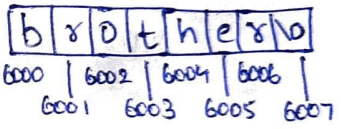
(III) ++p -> c

↳ this operator

p->c ≡ (*p).c

-> has more precedence than ++





++p -> c

↳ this will be evaluated first

⇓

++(p->c) ≡ ++((*p).c)

p | 1010 | 7000

⇓

++((*1010).c)

1010 ≡ st+1

++(st[1].c)

*1010 ≡ *(st+1) ≡ st[1]

⇓

st[1].c = st[1].c + 1

= 3000 + 1x1B

st[1].c = 3001

(IV) p++ -> c ≡ (p++)->c

(1010++)->c

⇓

(1010->c)

p->c ≡ (*p).c ≡ 3001

p = p+1

p | 1010 | 7000 ⟶ p | 1020 | 7000

printf("%s", p++->c) ≡ string starting at 3001 ⇒ etter

(V)

*++p->c p [1020] 7000

this will be evaluated first

*++(p->c) p->c ≡ (*p).c

= (*1020).c

*++s[2].c p->c = 4000 ≡ s[2].c

*(++s[2].c) ⟶ s[2].c = s[2].c + 1

= 4000 + 1×1B

*4001 s[2].c = 4001

u (character) printf("%c", *++p->c); Output → u

(VI)

p[0].i p [1020] 7000

1020 = st+2

*1020 = *(st+2) = st[2]

p[0] ≡ *(p+0)

*p

⇒ p[0].i ≡ (*p).i = (*1020).i

st[2].i = 6 output

(VII)

p->c ≡ (*p).c ≡ (*1020).c ≡ st[2].c = 4001

printf("%s\n", p->c); ≡ string starting at 4001 ⇒ ungle output

Q2 Which of the ~~following~~ choices given below would be printed when the following program is executed?

```
int a1[] = {6, 7, 8, 18, 34, 67};
int a2[] = {23, 56, 28, 29};
int a3[] = {-12, 27, -31};
int *x[] = {a1, a2, a3};
main() {
    f(x);
}
f(int **a) {
 I  printf("%d", a[0][2]);
 II  printf("%d", *a[2]);
 III  printf("%d", *++a[0]);
 IV  printf("%d", *(++a)[0]);
 V  printf("%d", a[-1][+1]);
}
```

a) 8, -12, 7, 23, 8

b) 8, 8, 7, 23, 7

c) -12, -12, 27, -31, 23

d) -12, -12, 27, -31, 56

Soln

a | 4000 | 8B
5000

a1

0	1	2	3	4	5
6	7	8	18	34	67

1000 1002 1004 1006 1008 1010

a2

23	56	28	29

2000 2002 2004 2006

a3

-12	27	-31

3000 3002 3004

x

a1 1000	a2 2000	a3 3000

4000 4008 4016

size of each elem = 8B (array of pointers)

(I) a[0][2]
*(a+0)
*a[2]
*(*a+2)
*(*4000+2)
*(1000+2x2)
*1004 = 8

a[0][2]
a1[2]
8

(II) *a[2]
*a3
*3000 = -12

(III) *++a[0]
*(++a1) ≡ *(a1+1) ≡ a1[1] = 7
*(1000+1x2)
*(1002) = 7

(IV) *(++a)[0]
evaluated first
*(a2[0])
23

a | 4000
5000
a | 4008
5000

(V) a[-1][+1]

266 SBI's sol

(II) *a[2]

a | 4000 |
5000

* *(a+2)

* *(4000+2)

* *(4000+2×8)

*(*4016)

*3000

-12

(III) *++a[0]

*(a+0)

*4000

*++3000

*1002

7

(IV) a[-1][+1]

*(a-1)[+1]

a | 4008 |
5000

((a-1)+1)

[(4008-1×8)+1]

*(*4000+1)

*(1002+1)

*(1002+1×2)

*1004

8

updated in step III

Lec-6 05/11/2020

Q Consider the following C-function

```
void foo(int n){
    while (n!=0){
        if (!(n&1))
            printf("*");
        n = n >>1;
    }
}
```

What will be printed for the function call foo(12)?

Output : * * equal to # of zeroes in 12

my attempt

foo(12)

12!=0 ✓

12 = 1100
1 = 0001
0000

!(12&1)

!0 = (1) ✓

Pf(*)

1100 >>1

(0110)

6!=0 ✓

!(6&1)

6 = 0110
1 = 0001
0000

!0 = 1 ✓

Pf(*)

0110 >>1

(0011)

3!=0 ✓

!(3&1)

3 = 0011
1 = 0001
0001

!(1) = 0

0011 >>1

(0001)

1!=0 ✓

!(1&1)

0001
0001
0001

!1 = 0

Pf(*)

0001 >>1

(0000)

0!=0 ✗

Q Suppose n and p are unsigned int variables in a C program, we wish to set p to $^nC_3$ if n is large which one of the following statements is most likely to set p correctly?

✗ (a) $p = n*(n-1)*(n-2)/6$

(b) $p = n*(n-1)/2*(n-2)/3$

(c) $p = n*(n-1)/3*(n-2)/2$

✗ (d) $p = n*(n-1)*(n-2)/6.0$

* since n is very large $n(n-1)(n-2)$ is going to be even greater
=> may cause overflow

* multiplication of two consequetive integers is always divisible by 2 $n(n-1)$
=> $n*(n-1)/2$ will have better performance and accuracy
and it may not be always divisible by 3

268 Q Consider the following program

```
void main() {
    char *x = "abc";
    char *y = "defgh";
    printlength(x,y);
}
```

```
void printlength(char *s, char *t) {
    unsigned int c=0;
    int len = ((strlen(s)-strlen(t)) > c) ?
                strlen(s) : strlen(t);
    printf("%d", len);
}
```

Recall that strlen() is defined in string.h as returning a value of type size_t, which is an unsigned int, the output of the program is 3

3 - 5 = -2 (converted to unsigned 2)

strlen(s) - strlen(t) > c

* it has to be an unsigned int as well

(unsigned int)

2 > 0 ✓ True

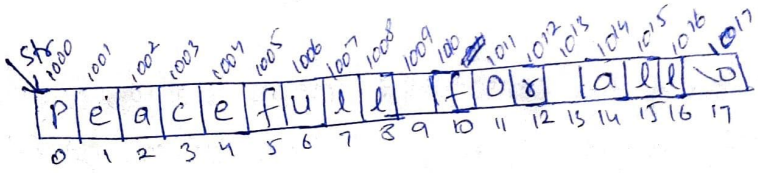
=> len = 3

p.f(3)

* strlen() returns unsigned int

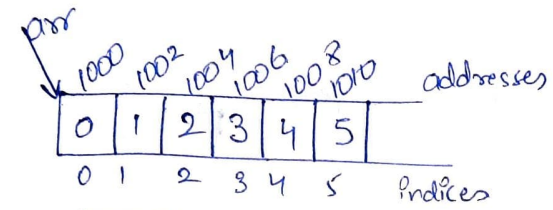
Q What would be the output of the following

```
void main() {
    char str[] = "peacefull for all ";
    clrscr();
    printf("%c\n", *(&str[2]));      *(1002) = 'a'
    printf("%s\n", str + 5);         1000 + 5x1 = 1005  "full for all"
    printf("%s\n", str);             "peaceful for all"
    printf("%c", *(str+2));          *(str+2) ≡ str[2] => 'a'
}
```



Q What is the output of the following code?

```
main() {
    int arr[] = {0,1,2,3,4,5};
    int i, *ptr;
    clrscr();
    for(ptr = arr+5; *ptr; --ptr) {
        printf("%d\n", *ptr);   ⟶ 5,4,3,2,1
}
```



arr | 1000 | 2000

*1002 = 1 !=0 ✓

pf(1)

*1000 = 0 0!=0 X

i: 8

ptr | ~~1010~~ 1008

*1010 = 5 !=0 ✓

pf(5)

*1008 = 4 4!=0 ✓

pf(4)

*1006 = 3 3!=0

pf(3)

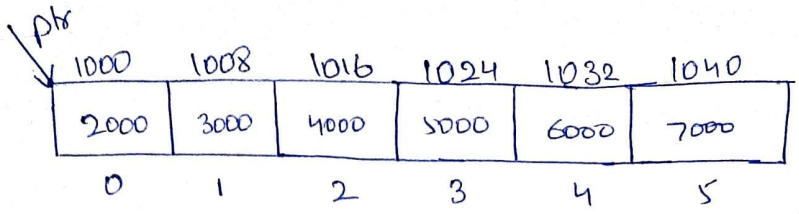
*1004 = 2 2!=0

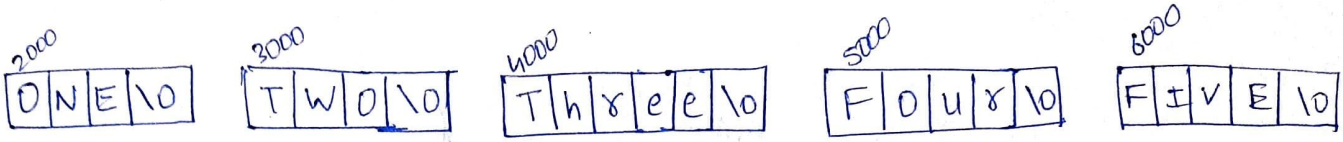
pf(2)

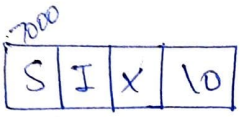
Q What is the output of the program?

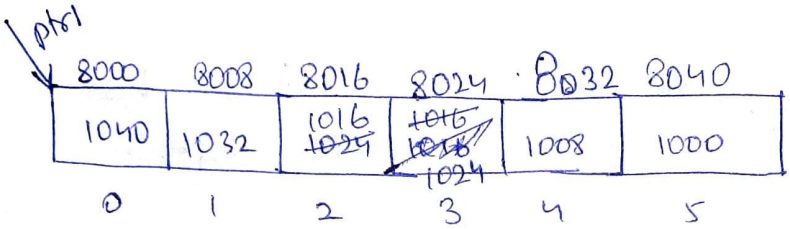
```
char *ptr[] = {"ONE", "TWO", "Three", "Four", "Five", "SIX"};
char **ptr1[] = {ptr+5, ptr+4, ptr+3, ptr+2, ptr+1, ptr};
char ***ptr2 = ptr1;
main() {
    int P=0;
    printf("%c", ++*++**ptr2++);
    printf("%s", ++**--*++ptr2+3);
    printf("%s", *ptr2[1]++3);
    printf("%s", --ptr2[-1][-1]+1);
```

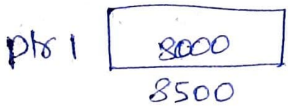
270 *size of pointer = 8B (assumed)











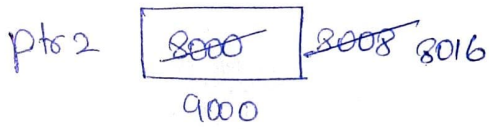
ptr + 5 = 1000 + 5x8 = 1040

ptr + 4 = 1000 + 4x8 = 1032

ptr + 3 = 1000 + 3x8 = 1024

ptr + 2 = 1000 + 2x8 = 1016

ptr + 1 = 1000 + 1x8 = 1008



++ * ++ * * ptr2++

(i) scan from left until operand encountered

++ * ++ * * (ptr2)++

operand

ptr2 = 8000

++ * ++ * * 8000 ++

both want 8000

* and ++ have same precedence but associativity is RTL

* ++ * ++ * * (8000++)

→ 8000 (because of post increment)

ptr2 = 8000 + *x8 = 8008 (will change)

++ * ++ * (*8000)

→ 1040

++ * ++ (*1040)

→ 7000

++*(++7000)

↓

++(*7001)

↓

(++I)

↓

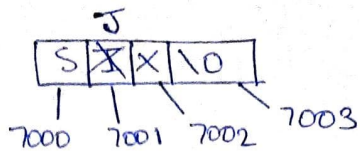
(char)(73+1)

(char)(74)

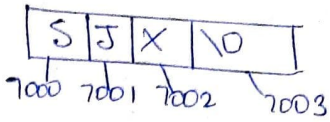
⇓

(J)

7000 → ptr[5]



ASCII of I = 73



++ * -- * ++ ptr2 + 3

↓

++ * -- *(++8008) + 3

↪ ptr2 = 8008 + 1×8 = 8016 (will change)

++ has higher precedence than +
(unary) (binary)

++ * --(*8016) + 3

↓

++ *(--1024) + 3

↓

++(*1016) + 3

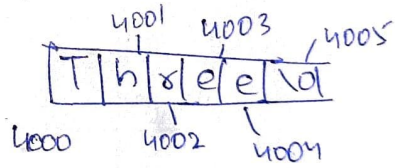
(++4000) + 3

↪ 4000 + 1×1 = 4001

4001 + 3 = 4004

* has higher precedence than +
(unary) (binary)

-- has higher priority than +
(unary) (binary)



print everything starting from 4004 => e output

---

* ptr2[1] +++3

↓

*(8016[1]) +++3

↓

((8016+1)) +++3

↓

*(*8024) +++3

↓

*(1016++) +3 → change to 1024

↓

(*1024) + 3

4001 + 3 = 4004 print all starting from 4004 => e output

[] has higher precedence than *

272

```
--ptr2[-1][-1]+1
  ↓
--(*(8016-1))[-1]+1
  --(*8008)[-1]+1
    ↓
  --(1032[-1])+1
    ↓
  --(*(1032-1))+1
  --(*1024)+1
  (--5000)+1
    ↓
  4999+1 = 5000   print everything starting from 5000
```

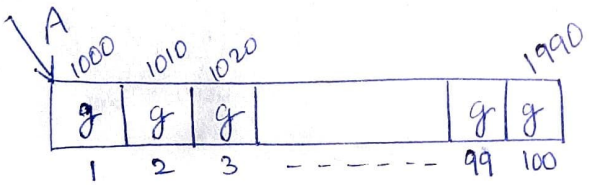
=> Four output

Lec-7 08/11/2020

Arrays

Q Let A[1...100] be an array where each element of size 10 Bytes and base address of array is 1000 then find address of element A[79].

Sol^n



g → Garbage value

* Total elements = $j - i + 1$

Address (A[79]) = BA + (x − i) × S

(Base Addr; x = index of element / whose addr to find; S = size of one element; i = 1, j = 100)

= 1000 + (79 − 1) × 10

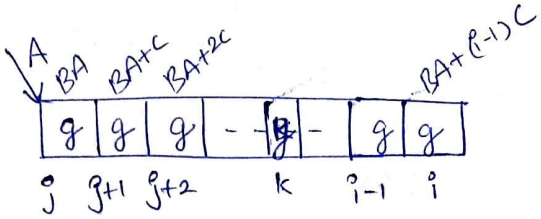
= 1000 + 780

Addr (A[79]) = 1780

* elements before j^{th} element = $j - i$

Q let A[j...i] be an array where each element of size C, base address of array BA, then find address of A[k].

Soln



$Addr(A[k]) = BA + (k-j) \times C$

↳ cross all elements before k

Total elements = $i-j+1$

Total size = total elements x size of one element

$= (i-j+1) \times C$

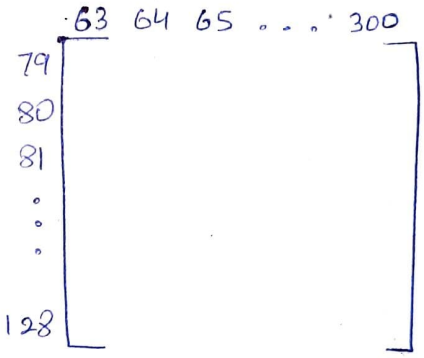
Q Let A[79...128, 63...300] be an array where each element of size 5 bytes, base address of an array is 500 and array stored in row major order, then find address of element A[125][233]

Soln

starting row, ending row

A[79...128][63...300]

starting column, ending column



# of rows = 128 - 79 + 1

R = 50

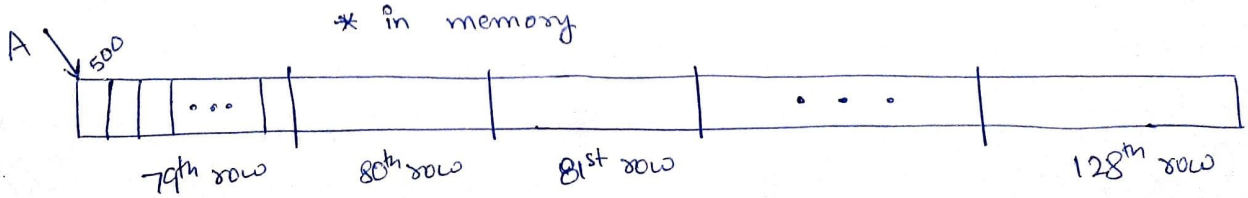
# of cols = 300 - 63 + 1

C = 238

1 row = 238 elements

1 col = 50 elements

* inside computer memory every array is stored as 1D array
* in row major, store row by row
* in col major, store col by col



274 *Row Major

Addr(A[125][233]) = BA + [(i - SR) × N_R + (j - SC)] × S

(Starting Row, # of elements in one row, size of one element, # of elements, (Starting col))

= 500 + [(125 - 79) × 238 + (233 - 63)] × 5

(Skipping rows)

R = 50 C = 238

N_R = 238 S = 5B

SR = 79 SC = 63

= 500 + (46 × 238 + 170) × 5

= 500 + (10948 + 170) × 5

= 500 + 11118 × 5

= 500 + 55590

Addr (A[125][233]) = 56090

```
  1 3
  2 4
  238
  x46
 1428
 952x
10948
 +170
11118
x    5
55590
  500
56090
```

* Column Major

(# of elements in one column)

Addr (A[125][233]) = BA + [(j - SC) × N_C + (i - SR)] × S

SR = 79 SC = 63

N_C = 50 R = 50

C = 238 S = 5B

= 500 + [(233 - 63) × 50 + (125 - 79)] × 5

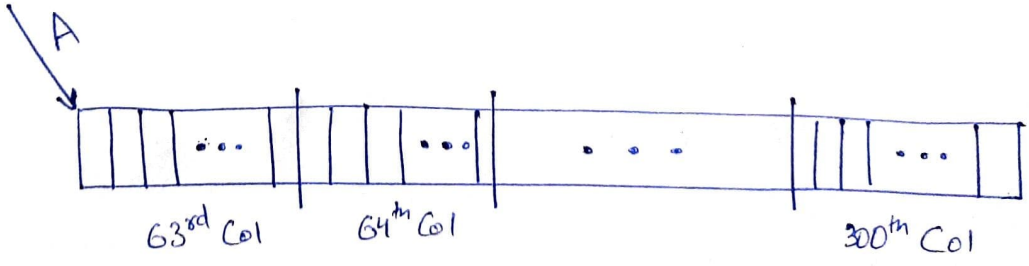
= 500 + [170 × 50 + 46] × 5

= 500 + (8500 + 46) × 5

= 500 + 8546 × 5

= 43230

```
  170
 x 50
 ⇓
17 x 5 = 50 + 35
170 x 50 = 8500
8500 + 46 = 8546
 2 23
 8546
  x  5
42730
  500
43230
```



Q Let A be a 2 dimensional array declared as

Int A[1...10, 1...15];

assuming that each integer takes one memory location ($\Rightarrow S=1$), the array is stored in row major order and the first element of the array is stored at the location 100, what is the address of the element A[i][j]

(a) $15i + j + 84$ (Row major) ✓

(b) $15j + i + 84$

(c) $10i + j + 89$ (Col major)

(d) $10j + i + 89$

my soln

$$A[i][j] = BA + [(i - SR) \times N_R + (j - SC)] \times S$$

$$= 100 + [(i-1) \times 15 + (j-1)] \times 1$$

$$= 100 + (15i - 15 + j - 1)$$

$$= 100 + 15i + j - 16$$

$$= 15i + j + 84$$

$R = 10 - 1 + 1$

$R = 10$

$C = 15 - 1 + 1$

$C = 15$

$N_R = 15$

$N_C = 10$

* Col major for A[i][j]

$$= BA + [(j - SC) \times N_C + (i - SR)] \times S$$

$$= 100 + [(j-1) \times 10 + i - 1] \times 1 = 100 + 10j - 10 + i - 1$$

$$= 100 + 10j + i - 11 = 10j + i + 89$$

Q In a compact single dimensional array representation for lower triangular matrices of size $n \times n$ [A[1...n][1...n]] non-zero elements of each row are stored one after another, starting from first row, the index of the $(i,j)^{th}$ element of the lower triangular matrix in this new representation is

(a) $i + j$ (b) $i + j - 1$ (c) $(j-1) + \frac{i(i-1)}{2}$ ✓

(d) $i + \frac{j(i-1)}{2}$

* Neither BA nor size of one element is not mentioned then

BA = 0 S = 1B (assumed)

example

A	1	2	3	4
1	a_{11}	0	0	0
2	a_{21}	a_{22}	0	0
3	a_{31}	a_{32}	a_{33}	0
4	a_{41}	a_{42}	a_{43}	a_{44}

Lower Triangular Matrix

* Normal representation = n^2 elements
* compact representation = $\frac{n(n+1)}{2}$ elements

* row major order - Lower ∆ar matrix

A	0	1	2	3	4	5	6	7	8	9
	a_{11}	a_{21}	a_{22}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}	a_{44}
	row1		row2		row3			row4		

$$Addr(A[i][j]) = BA + \left[\frac{i(i-1)}{2} + (j-1)\right] \times S$$

n = 4 $\frac{n(n+1)}{2} = \frac{4\times5}{2} = 10$ elements

cross all rows before i^{th} row => # of rows before i = i-1

# of elements before $i = \frac{(i-1)(i-1+1)}{2} = \frac{i(i-1)}{2}$

in our case BA = 0, S = 1

$$\Rightarrow Addr(A[i][j]) = (j-1) + \frac{i(i-1)}{2} \Rightarrow \text{option (c)}$$

example

	1	2	3	4
1	a_{11}	a_{12}	a_{13}	a_{14}
2	0	a_{22}	a_{23}	a_{24}
3	0	0	a_{33}	a_{34}
4	0	0	0	a_{44}

Upper Triangular Matrix

A	0	1	2	3	4	5	6	7	8	9
	a_{11}	a_{12}	a_{13}	a_{14}	a_{22}	a_{23}	a_{24}	a_{33}	a_{34}	a_{44}
	row1				row2			row3		row4

$$Addr(A[i][j]) = BA + [$$

Cross all elements before i^{th} row => ~~elements first ... row~~

# of rows = i

elements in i^{th} row = k

$(i-1)^{th}$ row = k+1

$(i-2)^{th}$ row = k+2

$[i-(i-1)]^{st}$ row = k+i-1

(1st row)

i^{th} row = n-i-1 elements

total elements = $\frac{n(n+1)}{2}$

elements from i^{th} row onward = i

1 2 3 -- i

n-i-1 $\frac{(n-i)(n-i-1)}{2}$

$$\frac{n(n+1) - (n-i)(n-i-1)}{2}$$

$$= \frac{n^2+n - (n^2 - ni - n - in + i^2 + i)}{2}$$

$$\frac{2n(1+i) - i(i+1)}{2}$$

$$\frac{(2n-1)(i+1)}{2}$$

$$\frac{2n+2ni-i^2-i}{2}$$

n=4
i=3 $\frac{(2\times4-1)(4)}{2}$

$\frac{}{2}$ 7x2 = (14)

Lec-8 08/11/2020

Linked List

Consider the function defined below

```
struct Node {
    int data;                  // 2B
    struct Node *next;         // pointer of type struct Node, 8B   } => 10B
}

int f(struct Node *p) {
    return ((p==Null) || (p->next==Null) || ((p->data <= p->next->data) &&
                                              f(p->next)));
}
```

For a given Linked list p, the function f returns 1 iff

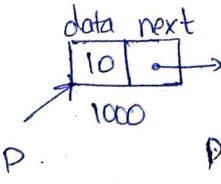
(a) The list is empty or has exactly one element

(b) The elements in the list are sorted in non-decreasing order of data (better option)

(c) The elements in the list are sorted in non-increasing order of data

(d) Not all elements in the list have same data value

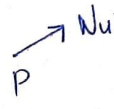
(i)



print(p) => 1000
print(*p) => .data = 10
.next = Null } -> shortcut => p->data=10, p->next=Null

f(p) =>
(i) p==Null ✗
(ii) p->next==Null ✓
(iii) not evaluated
} returns 1

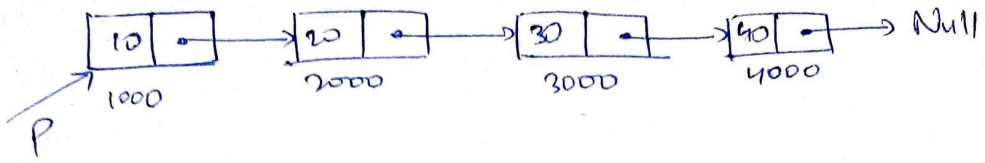
(ii)



empty Linked list f(p)
(i) p==Null ✓
(ii) and (iii) not evaluated } returns 1

p->data ≡ *p.data

if p is null (Linked List is empty) and *p is done then Segmentation error comes



print(p) => 1000 print (p->next->data) => 20

print(p->data) => 10

print(p->next) => 2000

(i) p == Null ✗ (ii) p->next == Null ✗ 0||0||1 => (1)

(iii) p->data ≤ p->next->data

10 ≤ 20 ✓ && f(p->next)

(1) && (1)

f(p->next) → (1)
2000

(i) p == Null ✗ (0) (ii) p->next == Null ✗ (0)

(1) (iii) p->data ≤ p->next->data

20 ≤ 30 ✓ && f(p->next)
3000

(1) && (1)

f(3000) → (1)

(i) p == Null ✗ (0) (ii) p->next == Null ✗ (0)

(1) (iii) p->data ≤ p->next->data

30 ≤ 40 ✓ && f(p->next)
4000

(1) && (1)

f(4000) → (1)

(i) p == Null ✗ (0) (ii) p->next == Null ✓ (1)

(iii) ~~p->data~~ not evaluated

* for increasing order and same values returns (1)

Q The concatenation of two list is to be performed in O(1) time, which one of the following implementation of a list should be used?

(A) Single Linked List => O(m)

(B) Double Linked List => O(m)

(C) Circular Double Linked List => O(1)

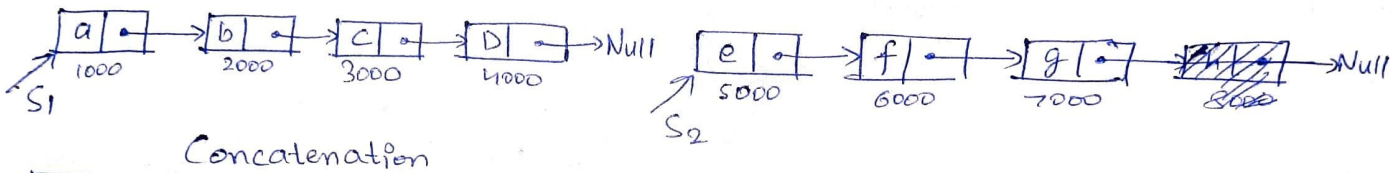
(D) Circular Single Linked List => O(m+n)

Concatenated of 2 Linked Lists

(i) Singly Linked List

$S_1 \rightarrow$ m size

$S_2 \rightarrow$ n size



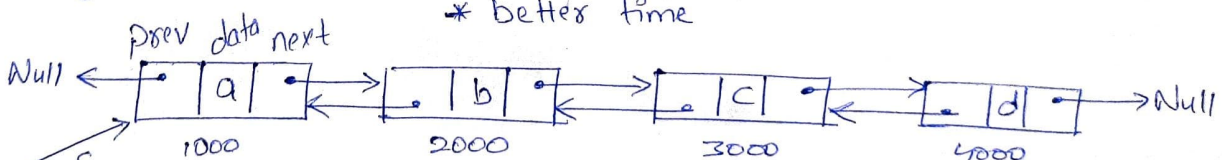
Concatenation

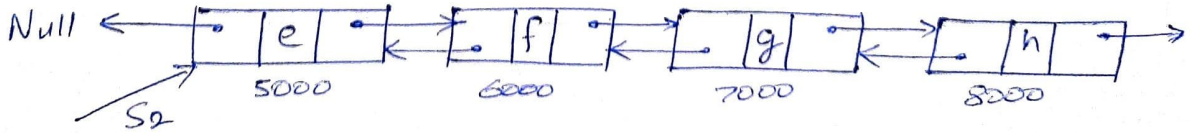
```
S = S1
while (S -> next != Null)
    S = S->next
S->next = S2
```

TC = O(m)

(ii) Doubly Linked List S_1 — m nodes S_2 — n nodes

* better time





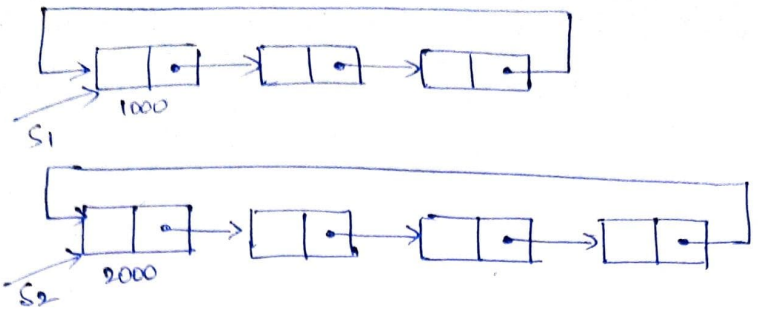
$S = S_1$ while ($S_1 \rightarrow$ next != Null) O(m) time

$S_1 = S_1 \rightarrow$ next

$S_1 \rightarrow$ next = S_2

$S_2 \rightarrow$ prev = S_1

(iii) Circular Linked List $S_1 - m$ $S_2 - n$



$S_0 = S_1$

traverse S_1 while ($S \rightarrow next \; != 1000$) $O(m)$

$S = S \rightarrow next$

$S \rightarrow next = S_2$

$S = S_2$

traverse S_2 while ($S \rightarrow next \; != 2000$) $O(n)$

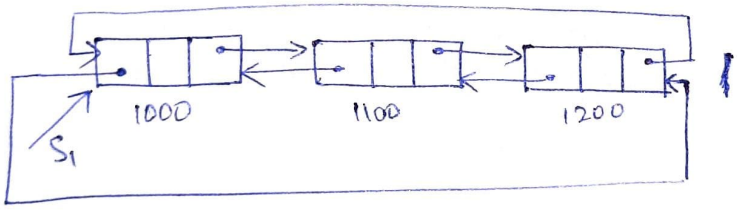
$S = S \rightarrow next$

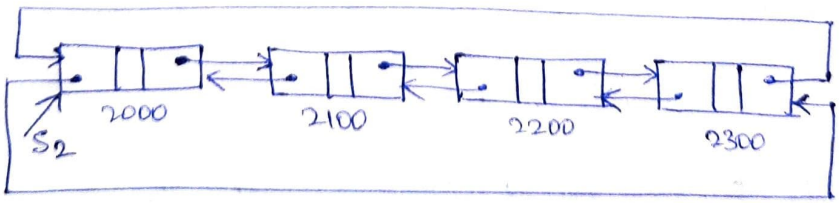
$S = S_1$

$O(m+n)$

(iv) Two Circular Doubly Linked List

$S_1 - m$ $S_2 - n$





$S_1 \rightarrow prev \rightarrow next = S_2$

$S_2 \rightarrow prev \rightarrow next = S_1$

Swap
$S = S_1 \rightarrow prev$
$S_1 \rightarrow prev = S_2 \rightarrow prev$
$S_2 \rightarrow prev = S$

$O(1)$ time

Q Consider the following C program fragment below

```
typedef struct node {
    int data;      ⟹ 2B
    node * next;   ⟹ 8B
} 10B
}node;
```

* created a type of node short for struct node

```
void join (node *m, node *n) {
    node *p = n;        ⟶ equivalent to struct node *p;
    while (p->next != Null) {
        p = p->next;
    }
    p->next = m;
}
```

assuming that m and n point to valid Null-terminated linked lists, invocation of join will

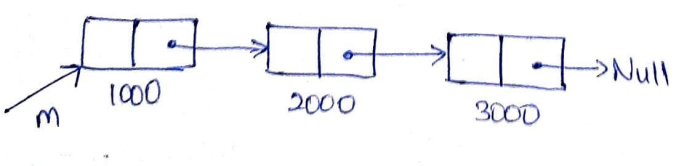
(A) Append list m to the end of list n for all the inputs * my ans -0.67

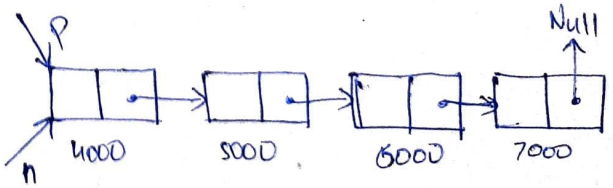
Better Option

(B) Either cause a null pointer dereference or append list m to the end of list n +2 * sir's answer

(C) Cause a null pointer ~~def~~ dereference for all inputs

(D) Append list n to the end of list m for all inputs





& - reference operator * - dereference operator

* if n is empty linked list ⟶ (*p).next => segmentation fault

282

Q N items are stored in a sorted double linked list, for a delete operation a pointer is provided to the record to be deleted. For a decrease key operation a pointer is provided to the record on which the operation is to be performed.

An algorithm performs the following operations on the list in this order: $\Theta(N)$ delete, $O(\log N)$ insert, $O(\log N)$ find and $\Theta(N)$ decrease key, what is the time complexity of all the operations put together?

(a) $O(\log^2 N)$

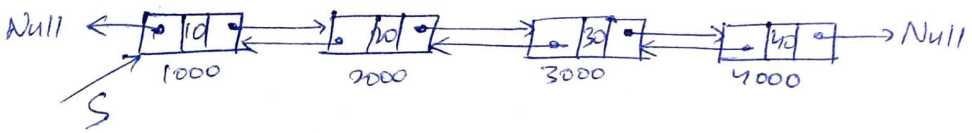
(b) $O(N)$

(c) $O(N^2)$

(d) $\Theta(N^2 \log N)$

* → pointer given

	Sorted DLL	
* N - Delete	$N \times O(1) \Rightarrow O(N)$ time	asymptotically $O(N^2)$ time
log N - Insert	$\log N \times O(N) \Rightarrow O(N \log N)$ time	
log N - Find	$\log N \times O(N) \Rightarrow O(N \log N)$ time	
* N - Dec. Key	$N \times O(N) \Rightarrow O(N^2)$ time	



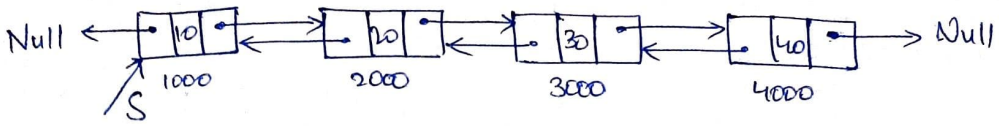
pointer is provided for record to be deleted

if p = 3000 (pointer to node with data, 30)

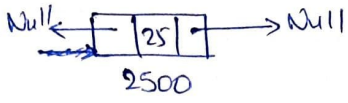
p → prev → next = p → next

p → next → prev = p → prev

} O(1) - Deleted one node



* Insertion →



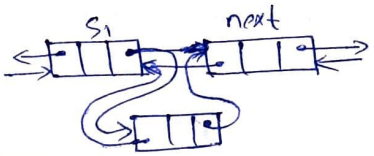
if we want to insert this node then we will have to insert properly in the sorted DLL

$S_1 = S$ new_node = Data (25)

while (S_1 -> next != null && S_1 -> data < new_node -> data) worst O(n)

$S_1 = S_1$ -> next

O(1) { new_node -> next = S_1 -> next
S_1 -> next = new_node
new_node -> prev = S_1
S_1 -> next -> prev = new_node nn



1 insertion

Total => O(n) time

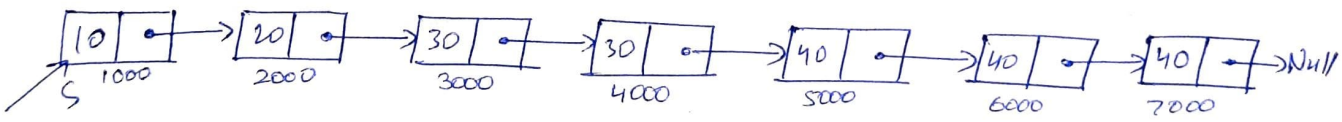
* Find a node => O(N) worst case

* Decrease Key → reduce data value

if we decrease the value of node with data 30 to 15 value then we will have to sort again → O(N) time in worst case.

* Comparisions with LHS elements

Q How to eliminate duplicate data in Sorted Singly ~~Sort~~ Linked List



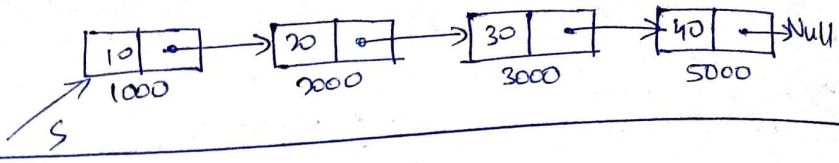
* duplicates will be adjacent => O(N) time required in every case

```
if (S == Null) || (S->next == Null)      // empty list / one element in list
  { return S; } S1 = S;
while (S1->next != Null) {
    if (S1->data == S1->next->data) {
        p = S1->next;        // we have to delete this
        S1->next = p1->next
        free(p); p = Null;
    }
```

→ if p doesn't point to Null it will keep pointing to freed memory location acting as a dangling pointer

Sorted SLL after removing duplicates

```
else {
    S1 = S1->next;
}
} → end of while loop
```



---

* in above question if list is unsorted

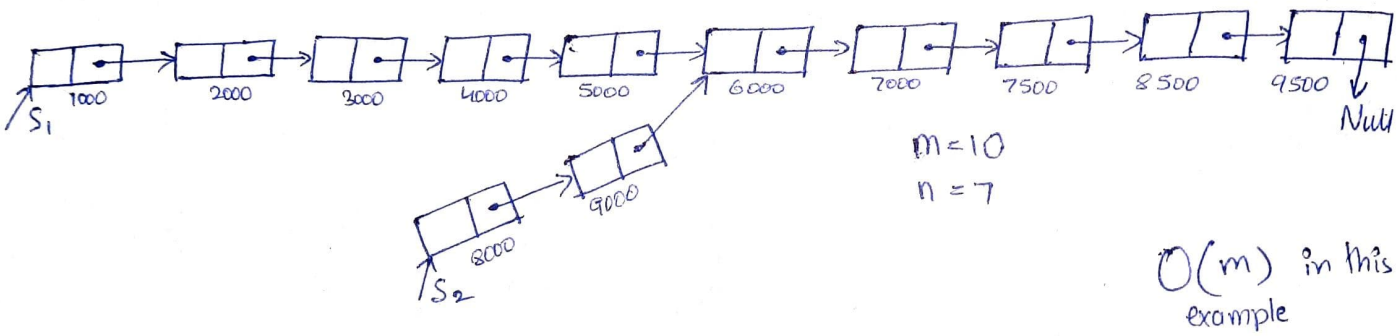
① sort using Merge Sort => O(NlogN)

② apply above algorithm => O(N)

O(NlogN) time

09/11/2020

Q How to find address of first intersecting node of given two singly linked lists each of length m and n respectively ?



m = 10
n = 7

O(m) in this example

* use while loop to find size of linked lists

① if (m > n)
    d = m - n
else
    d = n - m

m = 10 n = 7
d = 3

OR

d = (m > n) ? (m - n) : (n - m);

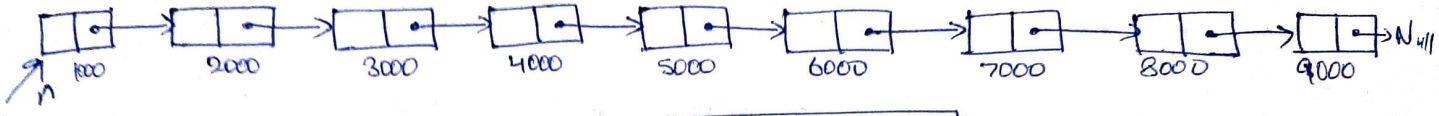
② while (d != 0) {
    S1 = S1->next;
    d = d - 1;
}

③ while (S1 != S2) {
    S1 = S1->next;
    S2 = S2->next;
}

④ return S1; or return S2;

TC = O(max(m,n))

Q Find the address of n^{th} node from the end of the linked list



TC. = O(n) every case

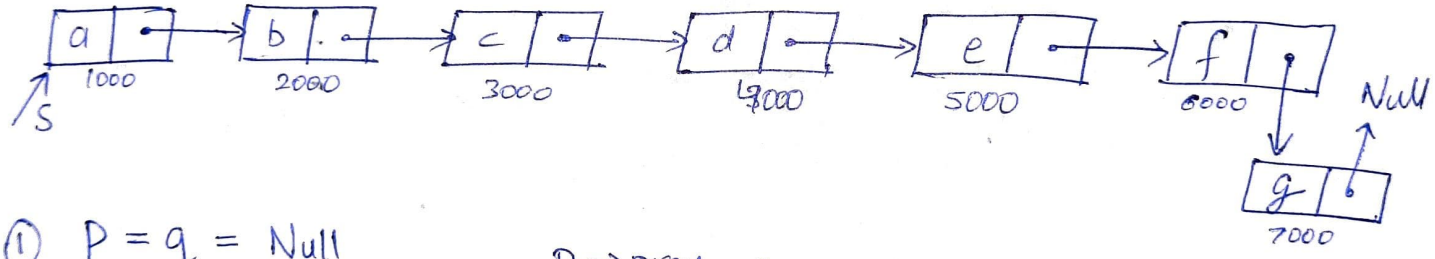
let n = 3

p = s

① while (n != 1) {
    n = n-1;
    s = s->next;
}

② while (s->next != Null) {
    s = s->next;
    p = p->next;
}
return p;

Q How to reverse the given single linked list ?



① p = q = Null

p → prev s → next q → current

② while (s != Null) { → whereever s is there, a node is there

    p = q;
    q = s;
    s = s->next;
    q->next = p;
}

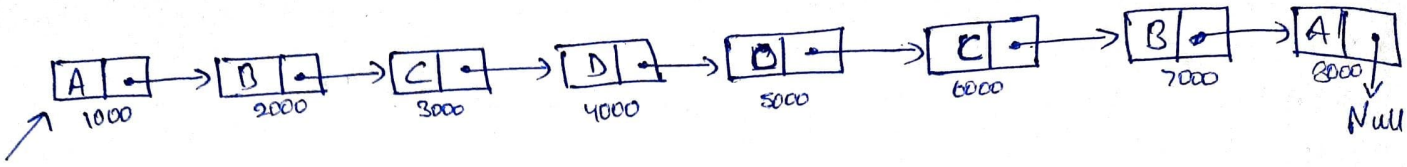
O(n) - every case (TC)

O(1) - extra space

③ return q; ← new head

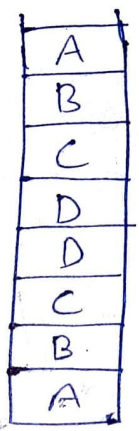
286 11/10/2020

Q How to verify whether given single linked list is Palindrom or not.




```
if (S == Null || S->next == Null)      // 0 or 1 node only
    return "Palindrome";

else { S1 = S
    while (S1 != Null) {                          O(n)
        push (S1->data);      // use stack
        S1 = S1->next;
    }
```

→ at the end of this loop → S_1 = Null S = 1000




```
    while (S != Null) {                           O(n)
        if (S->data == pop())
            S = S->next;
        else return "Not Palindrome";
    }

    if (S == Null)
        return "Palindrome";
    else
        return "Not Palindrome";
}
```

OR

* Another method

$n \Leftarrow$ (1) Find mid

$\frac{n}{2} \Leftarrow$ (2) ~~sort~~ reverse one half

$\frac{n}{2} \Leftarrow$ (3) compare both

$2n \Rightarrow$ O(n) = TC

O(1) = SC

~~Print~~

return "Palindrome";

Total = n+n = 2n

TC = O(n)

SC = O(n)

Lec-11 11/11/2020

** Trees (20-25 problems) (2-4 marks)

Important Topic

1. Binary Trees
2. Binary Search Trees
3. AVL Trees

Q The height of a tree is the length of the longest root to leaf path in it. The maximum and minimum number of nodes in binary tree of height 5 are

(A) 63 and 6 respectively ✓

(B) 64 and 5 respectively

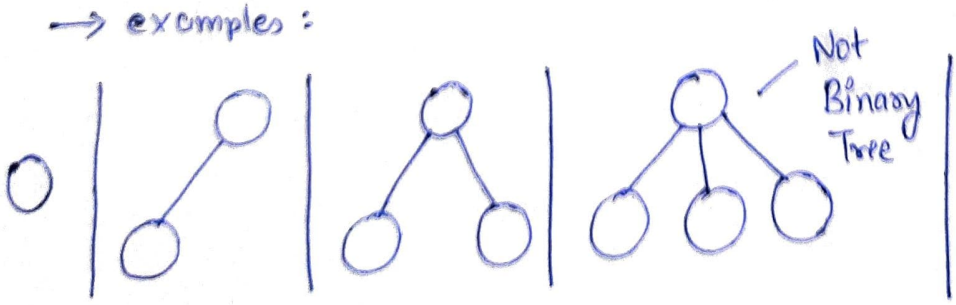
(C) 326 and 6 respectively

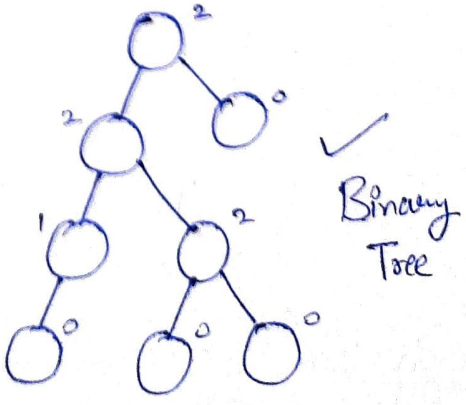
(D) 31 and 1 respectively

* Binary Tree → in the given tree, every node contains maximum 2 children

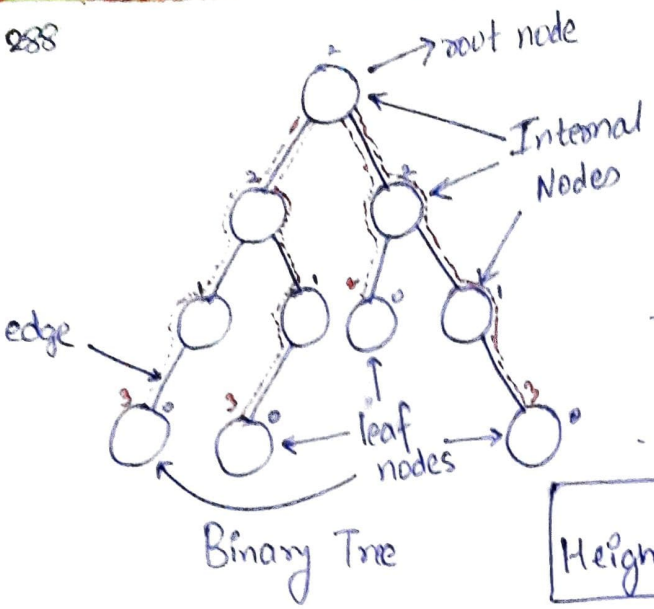
→ a tree with every node having at most 2 child nodes

→ examples :





288



* Total # of nodes
= 1 + 2 + 4 + 3
(L1) (L2) (L3) (L4)
= 10 Nodes

Height : longest Root to leaf node path

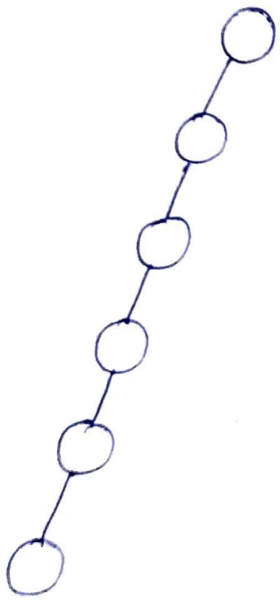
Height = # of Levels − 1

Here, height = max (3, 3, 2, 3)

height = 3

* in our question, height = 5
⇒ Levels = 5+1 = 6

* min BT with H = 5



N = 6 Nodes

* a tree in which every node has 2 child nodes
⇒ n^{th} level has = 2^{n-1} Nodes in it
⇒ total nodes = $2^n - 1$

* max BT with H = 5
⇒ every node with 2 child nodes
L1 = 2^0 Node
L2 = 2^1 Nodes
L3 = 2^2 Nodes
L4 = 2^3 Nodes
L5 = 2^4 Nodes
L6 = 2^5 Nodes

Total = $2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5$

GP, $r > 1$

$a = 2^0$, $r = 2^1$, $n = 6$

$$S_6 = \frac{a(r^n - 1)}{r - 1}$$

$$= \frac{2^0(2^6 - 1)}{2 - 1}$$

$$= 64 - 1$$

$S_6 = 63$ Nodes

Q A binary tree T has n-leaf nodes, the number of nodes of degree 2 in T is

(A) log n

✓(B) n − 1

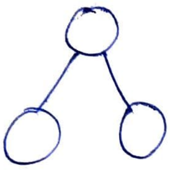
(C) n

(D) 2^n

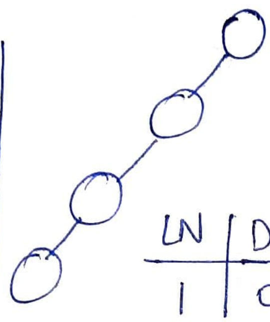
① take 3 examples of BT
② count LN, D-2 Nodes
③ Conclude with options

* in tree problems taking examples is better option

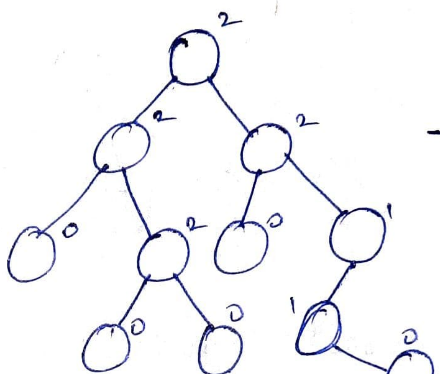
* Degree of a node = # of child nodes



LN	D-2
2	1



LN	D-2
1	0



LN	D-2
5	4

Q A binary tree T has 20 leaves, the number of nodes having 2 children is ___19___

n leaves => (n − 1) Degree 2 nodes

20 leaves => (20 − 1) Degree 2 nodes

---

0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7

1 Winner − (N − Rank − 1)) points

2 (N − R − 1))

last driver → N − (N − 1) = 1 point

3 Drivers

A B C

8 10 9

+3 3 3

+2 2 2

+1 1 1

14 16 15

3! = 6

1 A 3

2 B 2

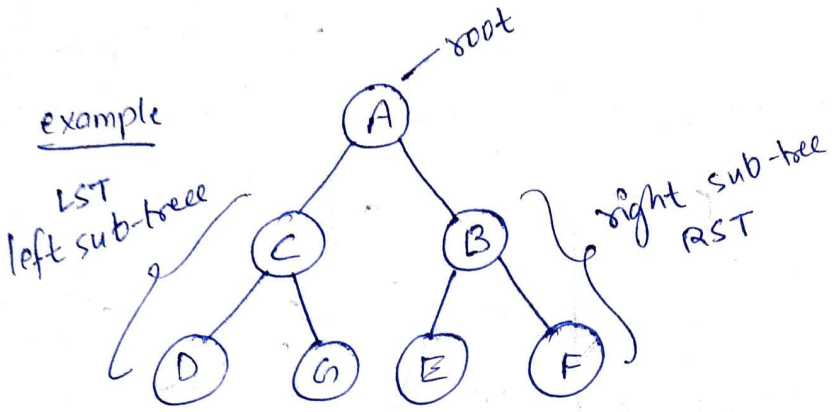
3

A 11 12 10

290

Q The post order traversal of a binary tree is 8,9,6,7,4,5,2,3,1. The inorder traversal of the same tree is 8,6,9,4,7,2,5,1,3. The height of the binary tree is _____

Soln



* Tree Traversal Techniques –

→ Preorder	Root	LST	RST	Data	Left	Right
→ Inorder	LST	Root	RST	Left	Data	Right
→ Postorder	LST	RST	Root	Left	Right	Data

* Preorder ⟹ Root at first for example binary tree

⟹ A C D G B E F ⟹ O(n)

* Inorder ⟹ D C G A E B F ⟹ O(n)

* PostOrder ⟹ D G C E F B A ⟹ O(n)

LST RST Root

Root at last

} TC

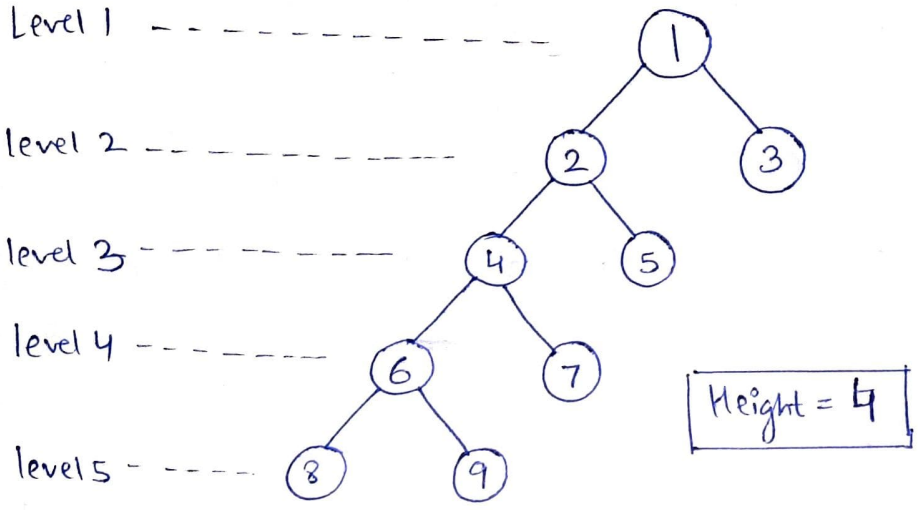
Solution

Postorder: 8 9 6 7 4 5 2 3 1 (Root) — LST: 8 9 6 7 4 5 2; RST: 3

Inorder: 8 6 9 4 7 2 5 1 3 — LST: 8 6 9 4 7 2 5; Root: 1; RST: 3

LST RST Root

LST Root RST



Height = 4

P 8 9 6 7 4 5 (2) root

I 8 6 9 4 7 (2) 5

LST RST

⇓

P 8 9 6 7 (4) Root

I 8 6 9 (4) 7

LST Root RST

⇓

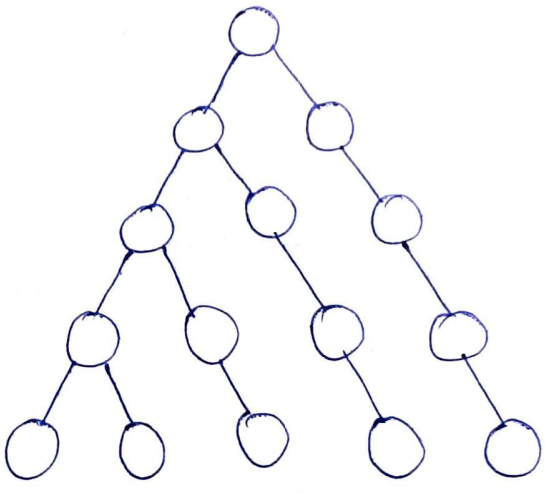
P 8 9 (6) Root

I 8 (6) 9

LST Root RST

pre-order: 1 2 4 6 8 9 7 5 3

Q Given the binary tree of the following form and having n-nodes, the height of the tree is



(A) $\Theta(\log n)$

(B) $\Theta(\sqrt{n})$

(C) $\Theta(n)$

(D) none

my soln In this form of tree, every level i has i number of nodes

if there are k levels, # of nodes = $1+2+3+\ldots+k = n$

$\frac{k(k+1)}{2} = n$ $\quad k^2+k-2n=0$ (quadratic) $\quad ax^2+bx+c$

$k^2+k = 2n$

$\Rightarrow k = \frac{-1 \pm \sqrt{1+4\times 2n}}{2} = \frac{\sqrt{8n+1}-1}{2} = k$

$k = O(\sqrt{n})$ asymptotically (levels)

$\Rightarrow k = O(\sqrt{n})$ asymptotically,

H = levels $-1 = \sqrt{n} - 1 = O(\sqrt{n})$

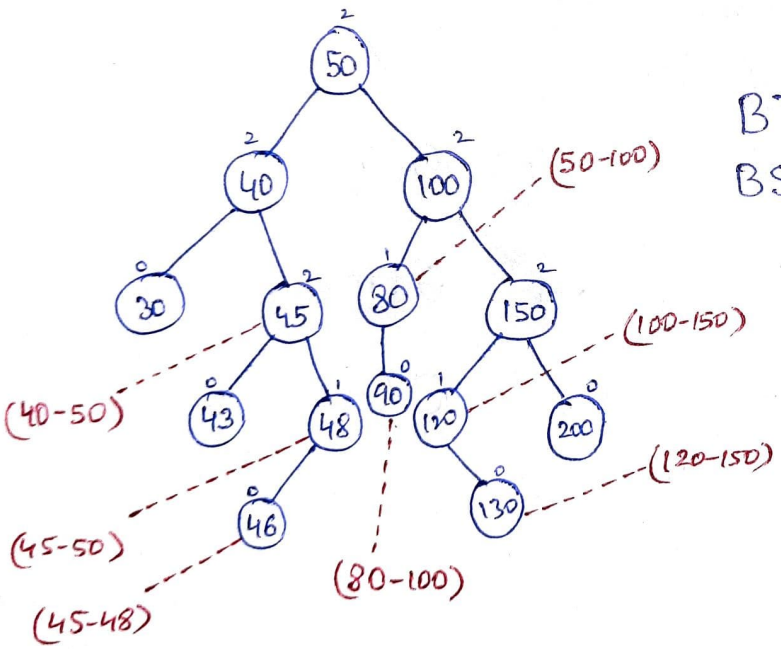
Q Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the in-order traversal sequence of the resultant tree?

(A) 7 5 1 0 3 2 4 6 8 9 (B) 0 2 4 3 1 6 5 9 8 7

(C) 0 1 2 3 4 5 6 7 8 9 (D) 9 8 6 4 2 3 0 1 5 7

* Binary Search Tree (BST)

→ a binary tree LST is smaller than Root and RST is greater than Root (at every node)



BT ✓
BST ✓

* Normally repeatition is not allowed in BST
* BST is used for the purpose of searching

* at every node

Left Child < Root < Right Child

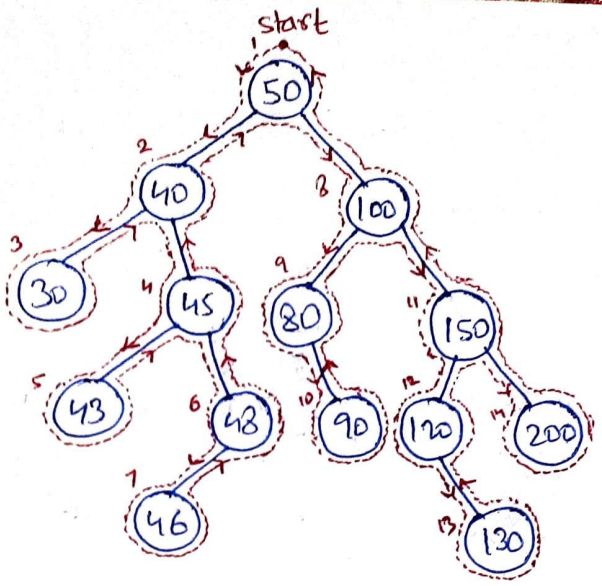
Preorder (Root Left Right) 50 40 30 45 43 48 46 100 80 90 150 120 130 200

Inorder (L Root R) 30 40 43 45 46 48 50 80 90 100 120 130 150 200

Postorder (LR Root) 30 43 46 48 45 40 90 80 130 120 200 150 100 50

all three => O(n) = TC (every case)

* Note : Inorder traversal of BST produces ascending order



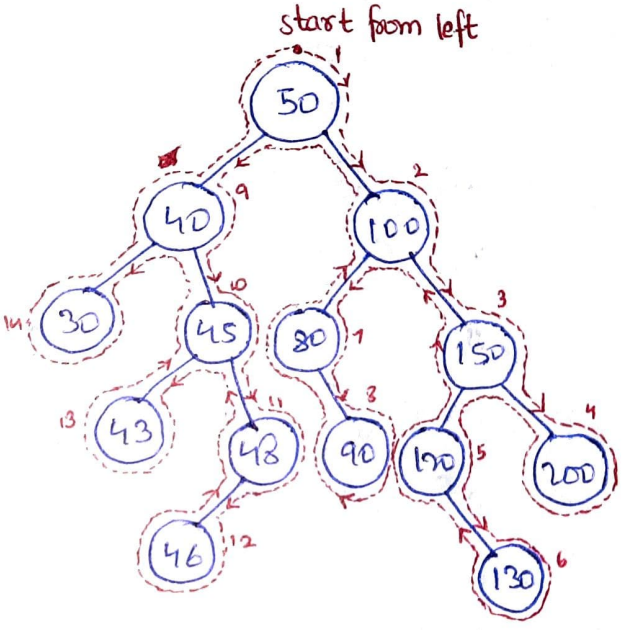
Preorder Root Left Right

50, 40, 30, 45, 43, 48, 46,
100, 80, 90, 150, 120, 130, 200

Inorder Left Root Right

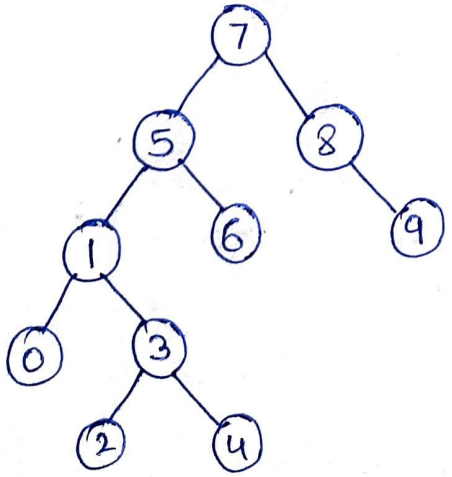
Post order Left Right Root

30, 43, 46, 48, 45, 40, 90,
80, 130, 120, 200, 150, 100, 50



12/11/2020

* BST for given problem



Preorder : 7, 5, 1, 0, 3, 2, 4, 6, 8, 9

Inorder : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Postorder : 0, 2, 4, 3, 1, 6, 5, 9, 8, 7

Q Draw the binary tree with node labels a, b, c, d, e, f and g for which the Inorder and postorder traversals result in the following sequences :

Inorder : a f b c d g e Postorder : a f c g e d b

My Soln

LRD LST RST
Post: a f c g e d (b)
In: a f (b) c d g e
LDR LST RST

⇓

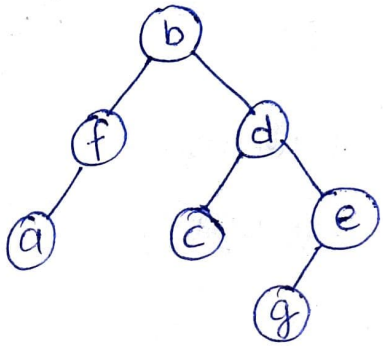
P: a f c g e (d)
I a f c d g e

⇓

P a f c g (e)
I a f g (e)

⇓

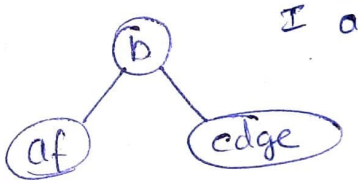
P a (f)
I a f



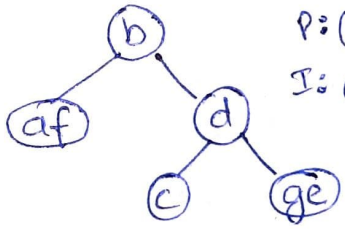
✓

Sir's Soln

P (af)(b)(cdge)
I afcged(b)

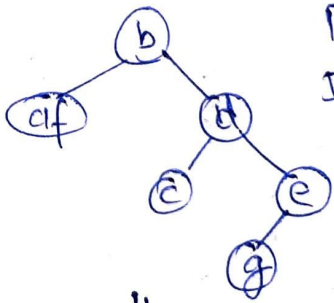


⇓



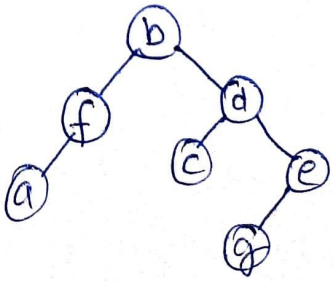
P: (c)(d)(ge)
I: afcge(d)(b)

⇓



P: (ge)
I: afcg(e)(d)(b)

⇓

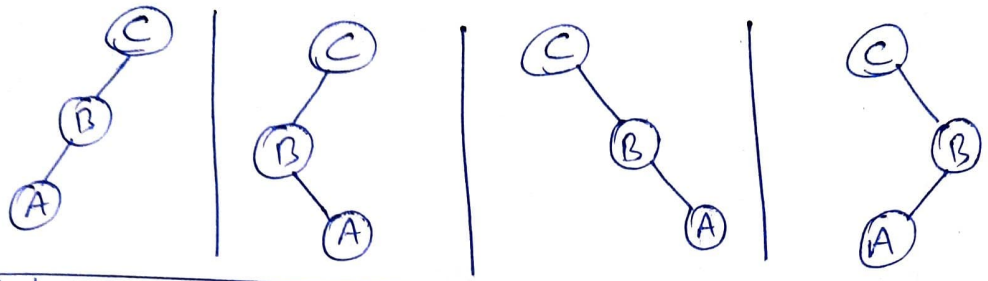


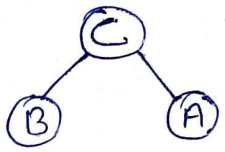
✓

Inorder : a f b c d g e → compulsory to be given along with either of these
Preorder : b f a d c e g
postorder : a f c g e d b

Q Draw all binary trees having exactly 3 -nodes labelled A, B and C on which preorder gives the sequence C, B, A

Preorder : C B A ROOT LST RST





* if preorder and inorder both were given then a unique BT will come.

21/01/21 Lec-12

The following points are for Binary Trees

* if preorder and in-order traversals are given then a unique binary tree is possible
* if post order and in-order traversals are given then one unique binary tree is possible
* only preorder traversal or only postorder traversal is given then multiple binary trees are possible
* if pre order and post order traversals are given => multiple binary trees are possible

Q A binary search tree contains the values 1, 2, 3, ..., 8
The tree is traversed in pre-order and the values are printed out. Which one of the following sequence is a valid output?

(A) 5 3 1 2 4 7 8 6
(B) 5 3 1 2 6 4 8 7
(C) 5 3 2 4 1 6 7 8
(D) 5 3 1 2 4 7 6 8

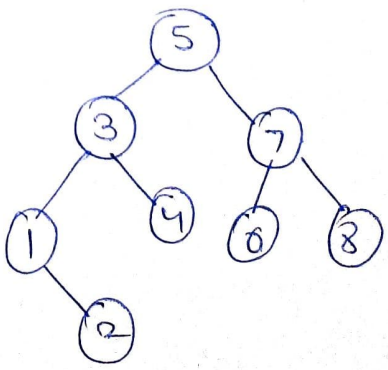
Inorder of BST => Ascending Order of elements

=> Inorder : 1 2 3 4 5 6 7 8

Construct BST with option (A) as Preorder

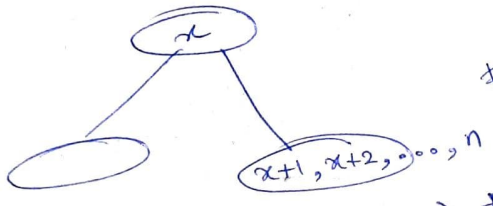
expected preorder : 5 3 1 2 4 7 6 8

Option D



(296) Q The numbers 1, 2, ..., n are inserted in a BST in some order. In the result, the right subtree of the root contains p-nodes. The first number to be inserted in the tree should be :

(a) p (b) p+1 (c) n-p (d) n-p+1



# of nodes in Right Subtree

$n - (x+1) + 1 = p$

$p = n - x$

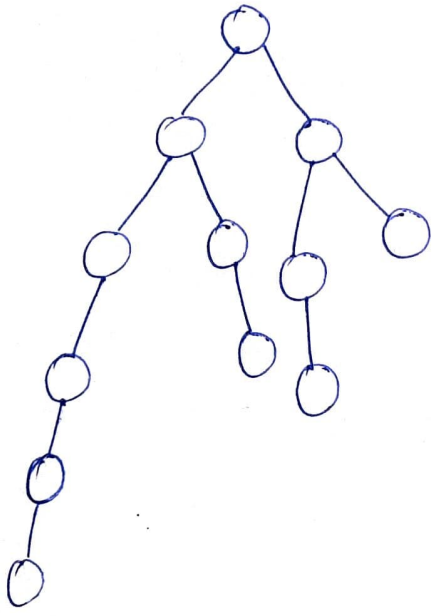
$x = n - p$

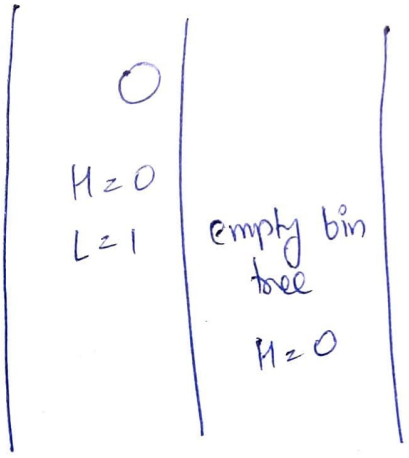
Lec-13 @ 26/01/2021

Q How to find height of a given binary tree?

Height : Longest distance from root node to leaf node

Height = Levels − 1





Binary tree is given
⇓ means
root is given
⇓
root node addr is given

if (root == Null)
empty bin tree

Height (root) = longest path from root to leaf

```
Height (root) ⟹ T(n)
{
① if (root == Null)
      return 0;
② if (root->left == Null && root->right == Null)
      return 0;
                                          ⟹ T(n/2)
③ L = Height (root->left);  // Height of Left Subtree      // minimum height = 1 at this point
  R = Height (root->right); // Height of Right Subtree
                                          ⟹ T(n/2)
  H = 1 + max (L, R);
  return H;
}
```

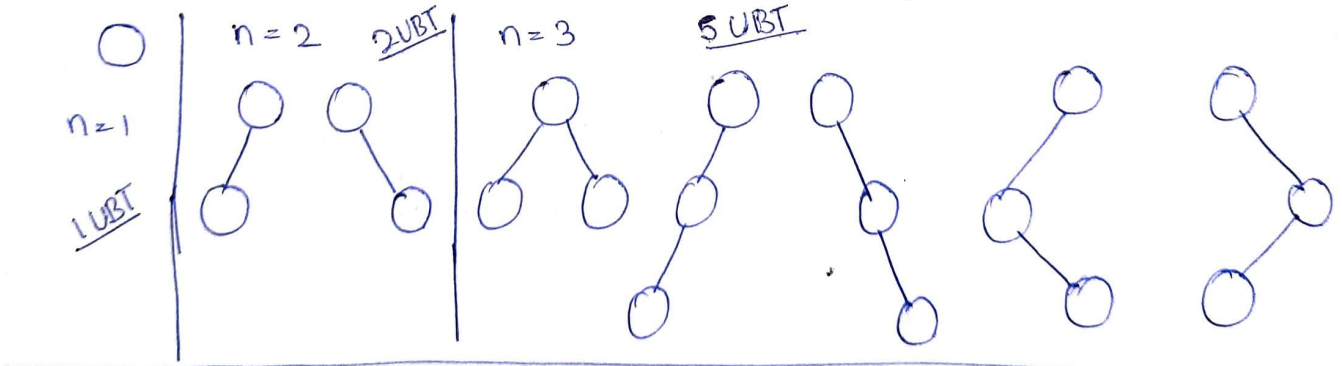
$T(n) = 2T(n/2) + C \implies O(n)$

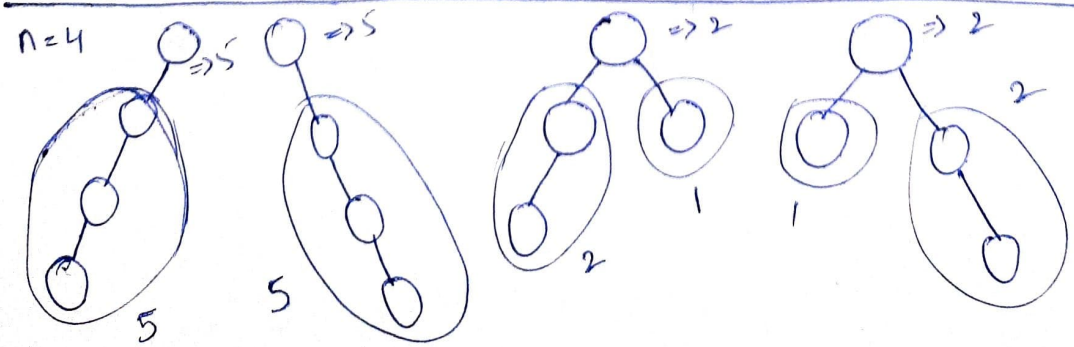
1 UBT = 1 BST

Q We are given a set of n-distinct elements and an unlabelled binary tree with n-nodes. In how many ways can we populate the tree with the given set so that it becomes a binary search tree?

a) 0 ~~b) 1~~ c) $n!$ d) $\frac{1}{n+1} \cdot {}^{2n}C_n$ ⇒ # of UBT possible for n number of nodes

Sol^n Unlabelled Binary Tree → # of Binary Trees possible



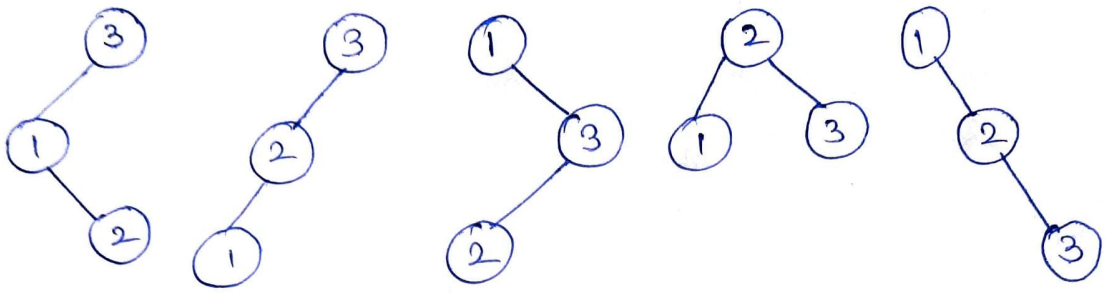


Total Unlabelled Binary Trees $= \frac{^{2n}C_n}{n+1}$, n – # of nodes

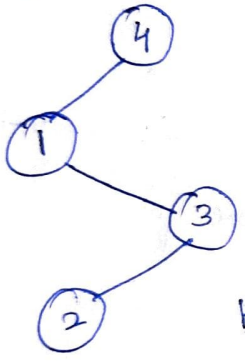
Binary Search Trees

# of Unlabelled Bin Trees = # of BSTs $= \frac{^{2n}C_n}{n+1}$

$n = 3 (1, 2, 3)$ 5 BST



Q Given a single unlabelled binary tree



$S = \{1, 2, 3, 4\}$

How many ways for elements of set S to be arranged in unlabelled binary tree to form BST ?

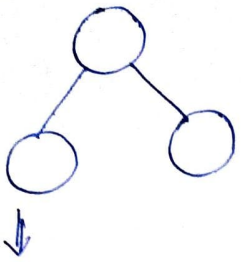
$\Rightarrow$ 1 BST

because # of UBT = # of BST

Q A weight balanced tree is a binary tree in which for each node the number of nodes in the left sub tree is at least half and at most twice the number of nodes in the right sub tree. The max possible height of such a tree on n-nodes is best described by which of the following ?

a) $\log_2 n$ b) $\log_{4/3} n$ c) $\log_3 n$ ✓ d) $\log_{3/2} n$

Weight Balanced Binary Tree



right subtree
# of elements = x

left subtree

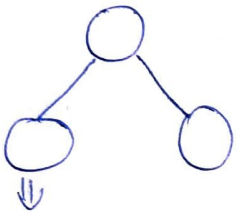
min max => # of nodes
$\frac{x}{2}$ 2x

# of nodes in left subtreee = y

$\frac{x}{2} \leq y \leq 2x$

where x is the # of nodes in right subtree

a) $\log_2 n$

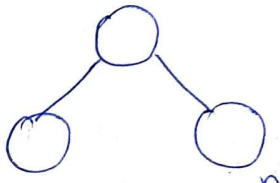


if Right Subtree has $\frac{n}{2}$ nodes

$\frac{n}{4} \leq y \leq 2 \times \frac{n}{2}$

let $y = \frac{n}{2}$ (within range)

$\Rightarrow$



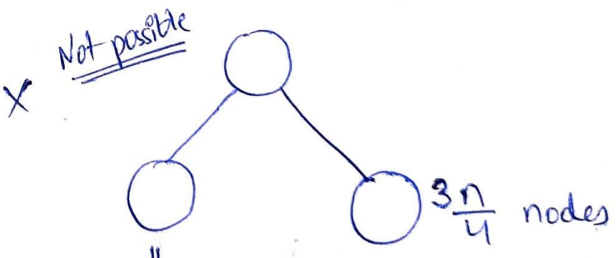
$n/2$ nodes $\quad$ $n/2$ nodes

$\log_2 n$ $\quad$ $\log_2 n$

$H = \max(\log_2 n, \log_2 n)$

$\boxed{H = \log_2 n}$

b)



$\frac{3n}{8} \leq y \leq \frac{3n}{2}$

$\Downarrow y = 3n/8$

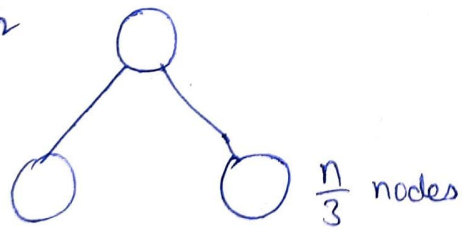
$\frac{3n}{8} + \frac{3n}{4} = \frac{6n+3n}{8}$

$= \frac{9n}{8}$

$\Downarrow$

greater than n

d) $\log_{3/2} n$



$\frac{n}{3}$ nodes

$\frac{n}{6} \leq y \leq \frac{2n}{3}$

$\Downarrow y = 2n/3$

$\log_{3/2} n$

$\Downarrow$

$\log_3 n$

$H = \max(\log_{3/2} n, \log_3 n)$

$H = \log_{3/2} n$

 Lec-14 26/01/2021

Q A BST is used to locate the number 43. Which of the following probe sequences are possible and which are not?

a) 61 52 14 17 40 43

b) 2 3 50 40 60 43

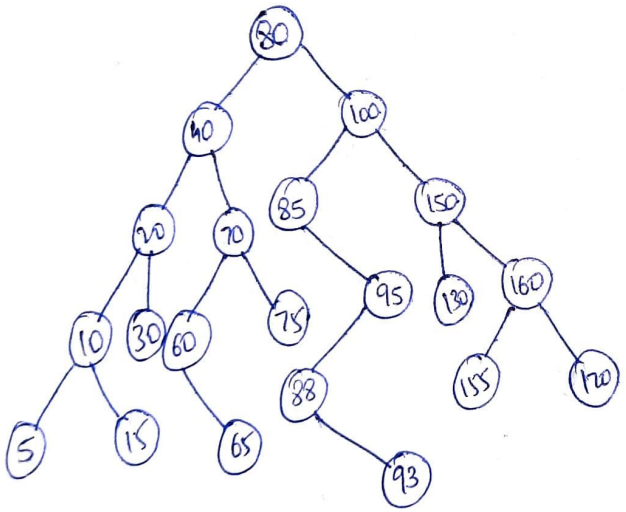
c) 10 65 31 48 37 43

d) 81 61 52 14 41 43

e) 17 77 27 66 18 43

Concepts

Binary Search Tree (BST) ⟹ greatest advantage is useful in Searching purpose



Search for element 65

65 == 80 ? × go left

65 == 40 ? × go right

65 == 70 ? × go left

65 == 60 ? × go right

65 == 65 ? ✓ Stop

80, 40, 70, 60, 65

probe sequence

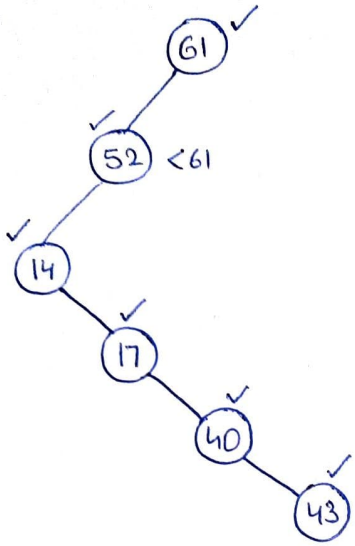
Searching

best case	O(1)	
worst case	O(n)	
avg case	O(log n)	

Solution

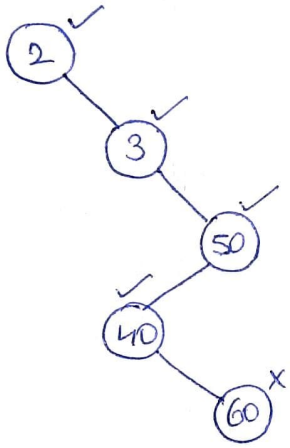
(A) 61 52 14 17 40 43 ✓ valid probe sequence

Valid



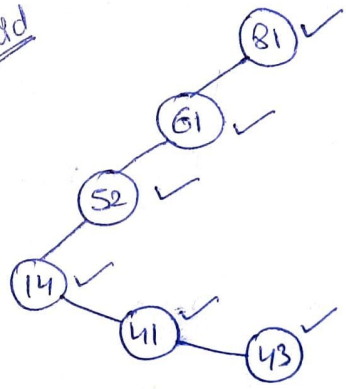
(B) 2 3 50 40 60 43 Invalid probe sequence.

Invalid



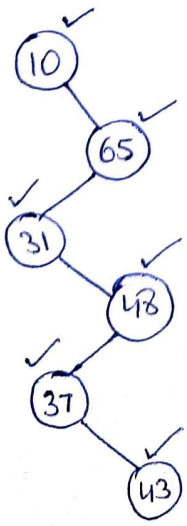
(D) 81 61 52 14 41 43

Valid



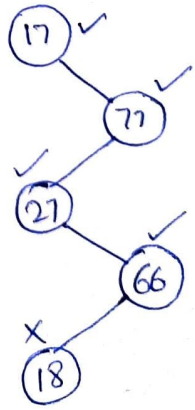
(C) 10 65 31 48 37 43

Valid



(E) 17 77 27 66 18 43

Invalid



302

BST (Search Time)	Bin Tree (Search Time)	AVL or Balanced BST
Best Case $\Rightarrow O(1)$	Best Case $\Rightarrow O(1)$	Best $\Rightarrow O(1)$
Worst $\Rightarrow O(n)$	Worst $\Rightarrow O(n)$	Worst $\Rightarrow O(\log n)$
avg $\Rightarrow O(\log n)$	Avg $\Rightarrow O(n)$	Avg $\Rightarrow O(\log n)$

Q A data structure is required for storing a set of integers such that each of the following operations can be done in $O(\log n)$ time, where n is the number of elements in the set.

1. Deletion of the smallest element $\Rightarrow$ maxheap = n, AVL Tree = log(n)
2. Insertion of an element if it is not already present in the set. $\Rightarrow$ max heap = log(n)+n, AVL = log(n)

Which one of the following data structure can be used for this purpose?

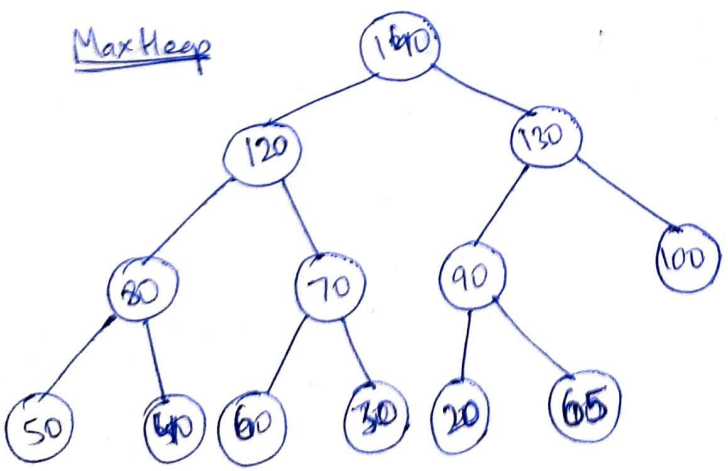
(A) A heap can be used but not a balanced BST

(B) A balanced BST can be used but not heap

(C) Both balanced BST and Heap can be used

(D) Neither balanced BST nor Heap can be used

* Heap means max heap
* Balanced BST means AVL Tree



Height = $\log_2 n$ (almost complete binary tree or complete binary tree)

Insertion
$O(\log n)$ worst & avg
$O(1)$ best case

Deletion in maxheap $O(\log n)$ worst and avg case
$O(1)$ best case

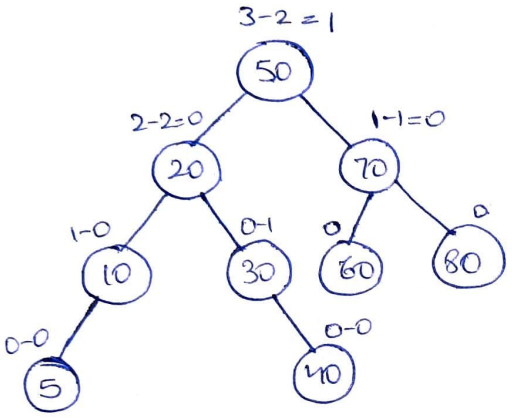
Find min element (one of the leaf)
$O(n)$ every case

Find max $\Rightarrow O(1)$ every case

Searching in max heap
O(n) worst and avg
O(1) best case

* best advantage is finding max element takes O(1) time in every case

AVL Tree or Balanced BST



$|L_H - L_R| \leq 1$

* difference in height of left subtree and right subtree is maximum 1
* Balancing factor at every node is $-1, +1$ or 0

* Searching
  O(1) Best Case
  O(log n) Worst & avg
* insertion
  O(log n) every case
* Deletion $x \Rightarrow$ O(log n) (every case)
  1) Search x
  2) Adjustment constant time for leaf nodes