Shri Ramdeobaba College of Engineering and Management.

Programme name :- Computer Science and Engineering.

Examination :- Winter 2020 ESE

Course Code :- CST 252

Course Name :- Data Structures and Algorithm

Roll no :- CSU20W-3006

Date :- 01/12/2020

Time :- 10 am to 12 pm

Q.1) a) Consider structure array.

```
struct Array
{ int *A;
  int length;
  int size;
}
```

Write a C program to delete all duplicate entries.

(b) Consider an array stores 30 students in 4 Subj. Row represents subjects and Column represents students. Base address = 2500. Lower bound = 0.

(a) Compute address of Marks [3][5] in row major.

(b) Compute address of marks [2][20] in Column major.

Q.2) (a) Consider a Stack ADT is already Created. write a function to check whether parenthesis are balanced or not.

(b) Design a C program for storing two queues stored in one array. Queue 1 grows from left to right and Queue 2 from right to left.

Q.3) (a) Write a C ~~program~~ function to delete all negative numbers from linked list.

(b) Write an algo. to interchange the K^{th} and $(K+1)^{th}$ node of a doubly linked list. Ensure that nodes are swapped and not just values.

Q.4) (a) A company maintains an array of employees which stores employee name, id and salary. Array is always kept in sorted order acc. to employee id. Design a C function which will search and display the details of a particular employee when ID is specified.

(b) Consider a hash table of size = 11. Use double hashing, insert keys 27, 72, 63, 42, 44, 36, 38 and 101 into table. $h1 = k \bmod 10$ and $h2 = k \bmod 8$.

Course Code :- CST252
Roll no :- CSU20W-3006
Rukhayyari

Q.5) (a) Construct a B+ tree of order 5.

80, 40, 15, 25, 30, 90, 35, 50, 60, 70

(b) Write C function to check whether root node is balanced or not.

Q.6) (a) Consider four cities. (1) New Delhi (2) Mumbai (3) Chennai (4) Banglore. Use diktra's algo to find shortest path to each node from New Delhi.

Flight no.	Origin	Destination	Cost
101	1	4	5
102	4	1	4
103	1	2	2
104	1	3	6
105	3	1	4
106	2	3	3
107	3	4	1
108	4	3	12

(b) Explain following terms :-
(a) Complete graph.
(b) Strongly connected graph.

Answers.

Q.1) (b) Word size = 2 → taken

(a) address of [3] [5] in row major.

$= \alpha + [i \times N + j] \times bpw$

$= 2500 + [3 \times 30 + 5) \times 2$

$= 2500 + [90 + 5) \times 2$

$= 2500 + 190$

$= 2690$

(b) address of [2] [20] in column major :-

$\alpha + [i + j \times M] \times bpw$

$= 2500 + [2 + 20 \times 4] \times 2$

$= 2500 + (2 + 80) \times 2$

$= 2664$

Q.2) (a)

(b)

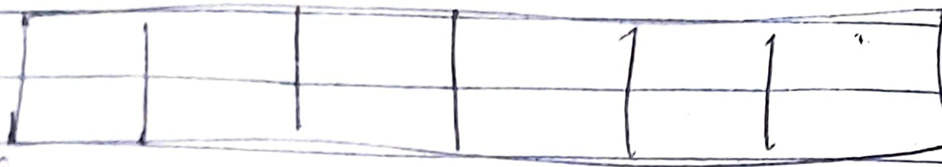
```
int check-balance (Stack sk, char a[]){
    int i = 0;
    while (i < len(a)){
        if (a[i] == ')'){
            if (isEmpty(sk)){
                return 0;
            } else if (peek(sk) == '('){
                pop(sk);
    }
```

```
else if (peek(sk) == ')'){
        return 0;
    }
} else {
        push(sk);
    }
    i++;
}
if (!isEmpty(sk)) {
        return 0;
    }
    return 1;
```

// isEmpty is a function which checks if
// stack is empty or not
// peek checks the top element
// pop and push removes and add respectively.

Q.2 b)

Queue 1 ⇒ ⇐ Queue 2



front1 →

rear1 →

~~top~~

← front2

← rear2

~~if~~ To insert in Queue 1

⇒ rear1++

To remove in Queue 1

⇒ front1++;

To insert in Queue 2

⇒ ~~top~~ rear2--;

To remove in Queue 1

⇒ front--;

Underflow condition :-

Queue 1 :- if front == -1 && rear == -1

}

~~int remove~~

```
void Insert (int opt, int x, int a[], int front1,
        int front2, int rear1, int rear2){
if ( opt = = 1) {
    // case of Queue 1
    if ((front1 == 0 && rear1 == len(a)-
        ~~|| front1 > rear1)~~ {
    {
        printf("Overflow\n");
    } else{
        a[++rear1] = x;
    }
}
```

```
else if (opt==2){
    // Case of Queue 2
    if (front2==n-1 && rear2==0){
        printf("Overflow\n");
    } else{
        q[--rear2] = x;
    }
}
void remove( int opt, int q[], int front1,
    int front2, int rear1, int rear2){
  if (opt==1){
  if (front1 > rear1){
        printf("Underflow\n");
    } else
        int ans=q[++front];
  }
```

```c
} else if (opt == 2) {
    if (front2 < rear2) {
        printf("Underflow\n");
    } else {
        int ans = a[--front2];
    }
}
~~return ans;~~
printf("%d", ans);
```

Q.3) (a)

→ // Assuming linked list created.

```c
void remove (struct Node** head) {
```

```
struct Node * temp = *head;
while (temp != NULL && temp -> data < 0){
    temp = temp->next;
    // cheeking the first positive integer.
}

struct Node * tp = * temp;
while ( tp != NULL) {
    if ( tp-> data > 0){
        create newnode (tp->data);
        temp->next = newnode;
        temp= temp->next;
    }
    tp = tp -> next;
}
```


Roll no: CSU 20W-3006
Relmaangasi

return temp;

}

Q. 37 (b)

```
void delete_kth( struct Node** head, int k){

    ~~while ( --k != 0)~~
    ~~while ( --k != 1)~~

    if ( head == NULL || head->next == NULL)
    { return head;
    }
    else {
        struct Node* node = *head;
```


Rellangoni

```
while ( --k != 1) {
        node = node -> next;
}

        node -> next = node->next->next->
                                    next;
return head;

}
```

Q.4) (b)

→ Size = m = 11

$h_1(k) = k \bmod 10$

$h_2(k) = k \bmod 8$

$h(k, i) = [h_1(k) + i * h_2(k)] \bmod M$

0	1	2	3	4	5	6	7	8	9	10

① $h(27,0) = [27 \% 10 + 0 \times 27 \% 8] \% 11$

$= 7 \% 11$

$= 7$

② $h(72,0)$

$= [72 \% 10 + 0 \times 72 \% 8] \% 11$

$= [2 + 0] \% 11$

$= 2$

Hash table :

0	1	2	3	4	5	6	7	8	9	10
-1	-1	72	-1	-1	-1	-1	27	-1	-1	-1

③ $h(63,0) = [63 \% 10 + 0 \times 63 \% 8] \% 11$

$= 3$

0	1	2	3	4	5	6	7	8	9	10
-1	-1	72	63	-1	-1	-1	27	-1	-1	-1

Course Code :- CST252
Roll no :- CSU20W-3006
Rilhargarsi

SHAKTI

DATE

(4) $h(42, 0)$

$= [42 \% 10 + 0 \times 42 \% 8] \% 11$

$= 2$

0	1	2	3	4	5	6	7	8	9	10
-1	-1	72	63	-1	-1	-1	27	-1	-1	-1

we can see, at index 2, 72 is present

$\therefore h(42, 1)$

$= [42 \% 10 + 1 \times 42 \% 8] \% 11$

$= [2 + 2] \% 11$

$= 4$

$\therefore$

0	1	2	3	4	5	6	7	8	9	10
-1	-1	72	63	42	-1	-1	27	-1	-1	-1

(5) $h(44, 0)$

$= [44 \% 10 + 0 \times 44 \% 8] \% 11$

$= 4$

We can see at index 4, 42 is present.

$\therefore h(44, 1)$

$= [44 \% 10 + 1 \times 44 \% 8] \% 11$

$= [4 + 4] \% 11$

$= 8$

$\therefore$

0	1	2	3	4	5	6	7	8	9	10
-1	-1	72	63	42	-1	-1	27	44	-1	-1

6) 36

$\Rightarrow h(36, 0)$

$= (36 \% 10 + 0 \times 36 \% 8) \% 11$

$= 6$

0	1	2	3	4	5	6	7	8	9	10
-1	-1	72	63	42	-1	36	27	44	-1	-1

(7) $h(38,0)$

$= (38\%10 + 0 \times 38\%8)\%11$

$= 8$

we can see at index 8, 44 is present.

$h(38,1)$

$= (38\%10 + 1 \times 38\%8)\%11$

$= (8 + 4)\%11$

$= 1$

0	1	2	3	4	5	6	7	8	9	10
-1	38	72	63	42	-1	36	27	44	-1	-1

~~At index 3, 63 is present~~

Course Code :- CST252

Roll no :- CSU20W-3006

Rellangang

~~$h(38, 2)$~~

~~$= (38 \% 10 + 2 \times 38 \% 8) \% 11$~~

~~$= 8$~~

(8) $h(101, 0)$

$= (101 \% 10 + 0 \times 101 \% 8) \% 11$

$= (1) \% 11$

$= 1$

At 1, 38 is present.

$h(101, 1)$

$= (101 \% 10 + 1 \times 101 \% 8) \% 11$

$= (1 + 5) \% 11$

$= 6$

At 6, 36 is present.

$h(101, 2)$

$= (101 \% 10 + 2 \times 101 \% 8) \% 11$

$= (1 + 10) \% 11$

$= 0$

0	1	2	3	4	5	6	7	8	9	10
101	38	72	63	42	-1	36	27	43	-1	-1

Hash table

Q.5) (a)

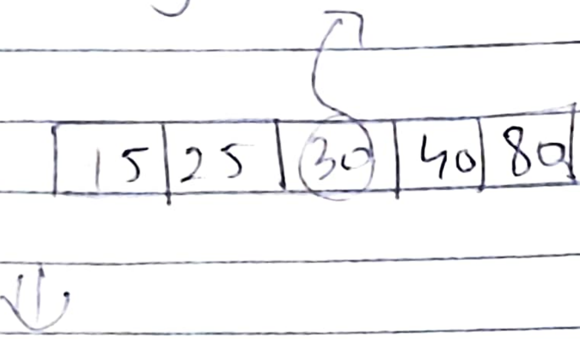
→ Order = m = 5

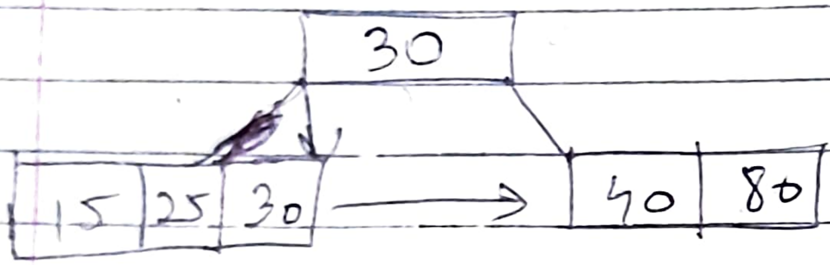
∴ Max no. of keys = m-1 = 4

(i) Inserting 80, 40, 15, 25

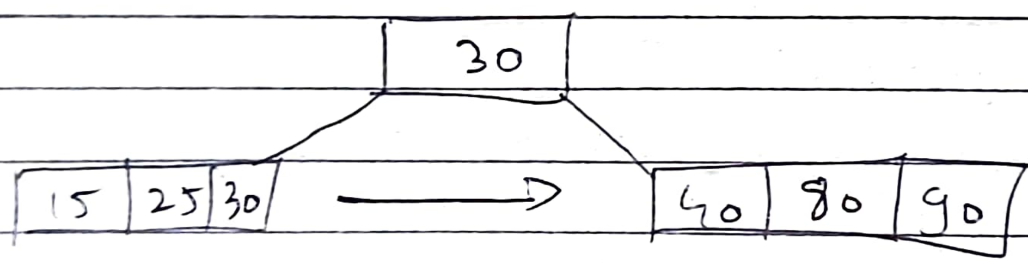
15	25	40	80

(2) Inserting 30

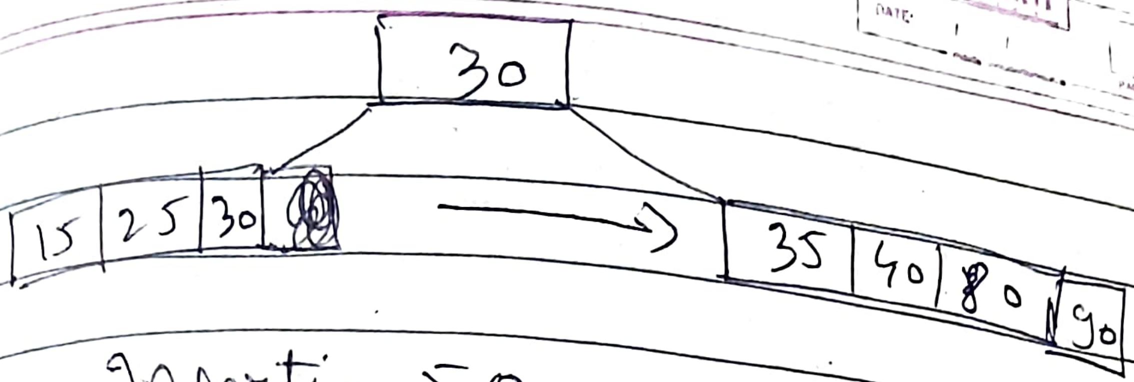




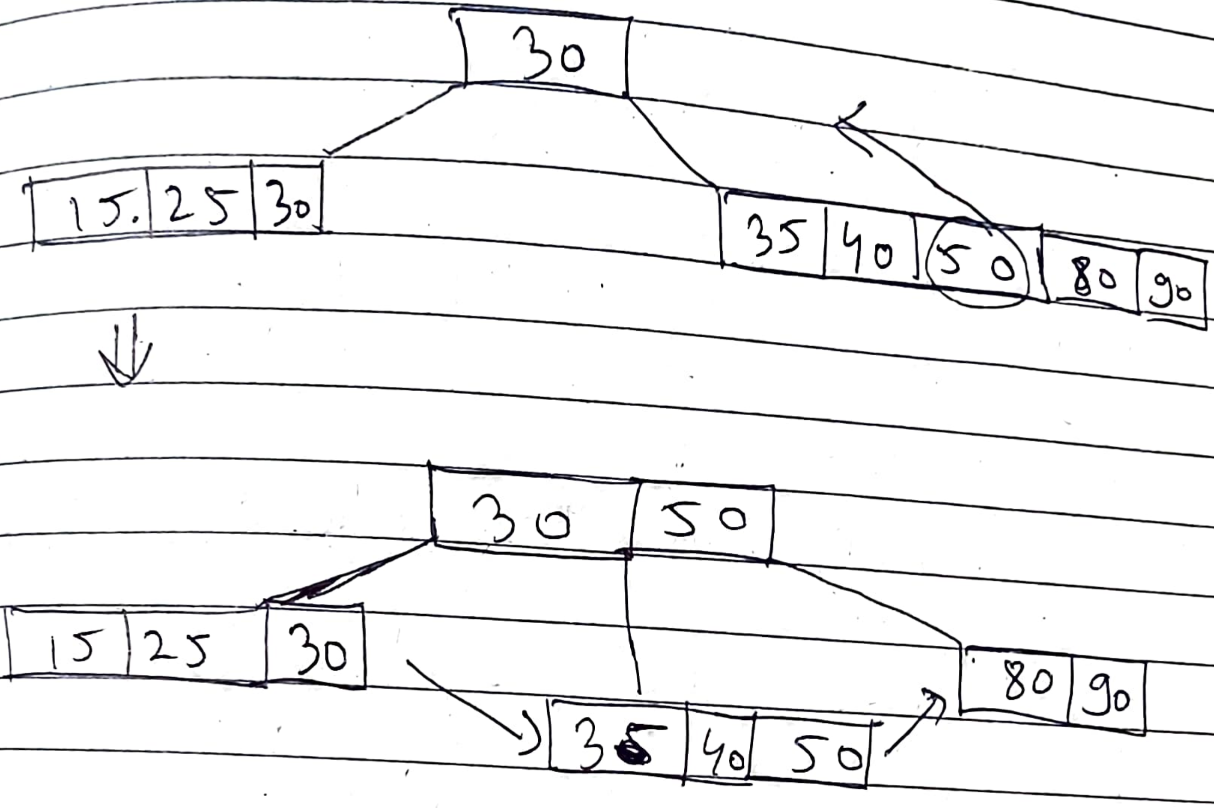
(3) Inserting 90,



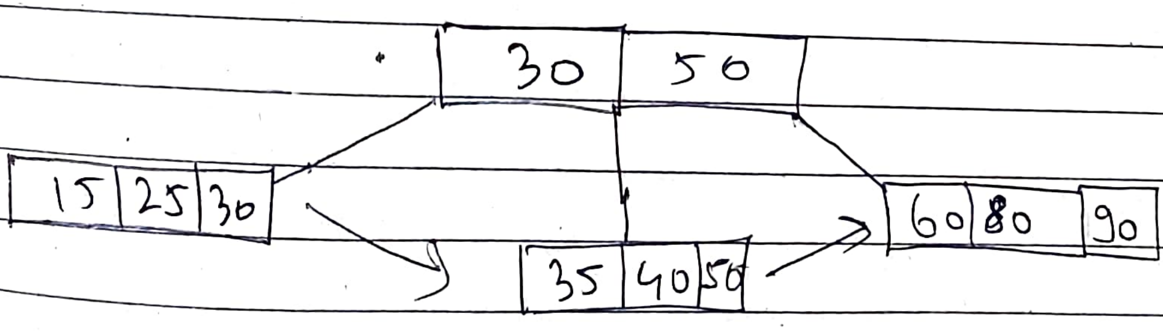
(4) Inserting 35,



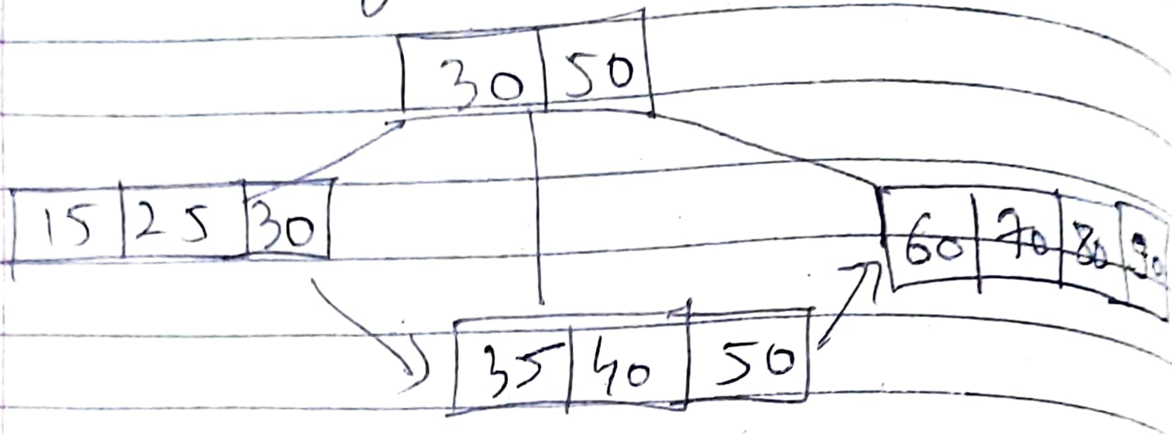
7 Inserting 50,



6 Inserting 60,



(i) Inserting 70,



Q.5) (b)

→

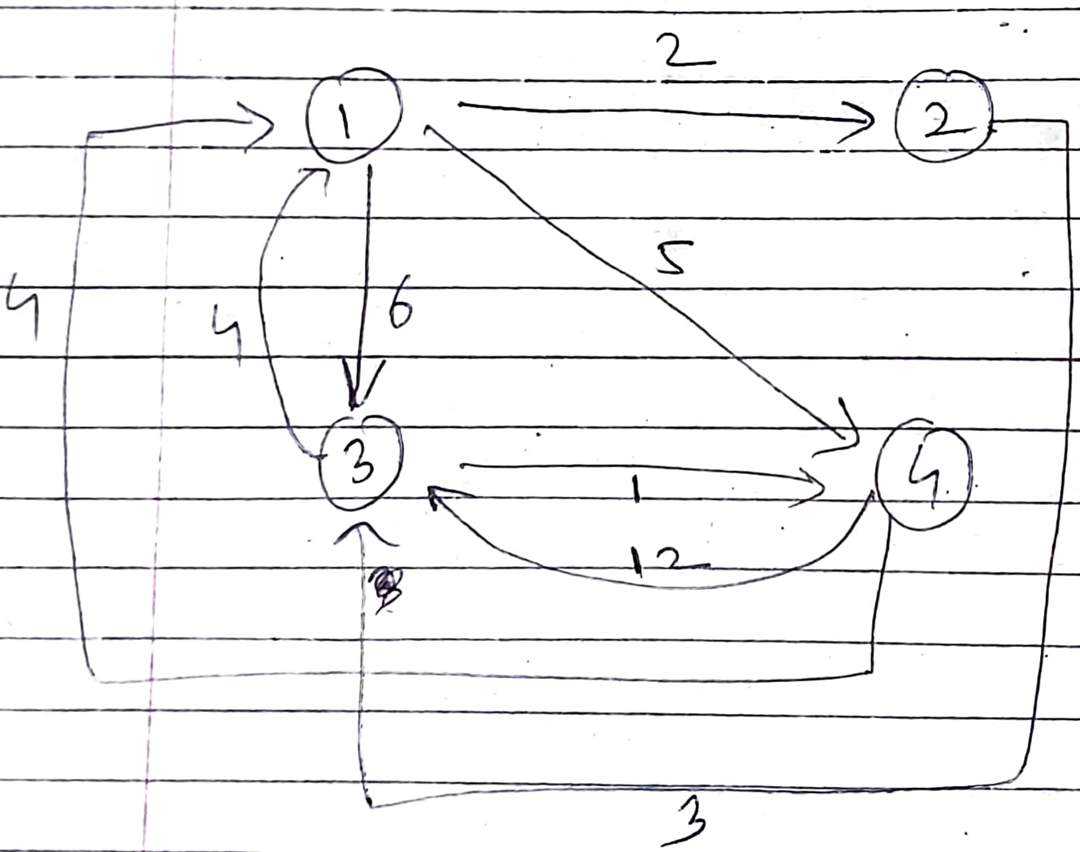
```
int height( treenode * root){
    if ( root ==NULL ) {
        return 0;
    }
        int l = height(root->left).
        int r = height(root->right).
```

```
return 1+max(l,r);

}

int checkbalance(treenode *root){
    if(root==NULL){
        return 1;
    }
        int l = height(root->left);
        int r = height(root->right);
    int ans = l-r;
    if(ans==1 || ans==-1 || ans==0){
        return 1;
    }
    return 0;
}
```

Q.6) (a)

Graph using the table will be

1 for New Delhi
2 for Mumbai
3 for Chenai
4 for Banglore.



Dijiktra's Algo from NewDelhi to each node.

Vertices	1	2	3	4
Prev. vertex	-	-	-	-
weight	0	-	-	-

Neighbouring nodes of 1 are.

2, 3 and 4.

∴ ~~Vertices~~	1	2	3	4
~~Prev. vertex~~	-	1	3	
~~Weight~~	0	2	6	

Vertices	1	2	3	4
Weight	0	2	6	5
Prev. vertex		1	1	1

But from 1 to 3 weight is 6 while min (6, path 123)

= min (6, 5) = 5

Vertices	1	2	3	4
weight	0	2	5	
Prev. vertex	-	1	2	

Minimum from Newdelhi to

Mumbai is 2
Chennai is 5
Banglore is 5

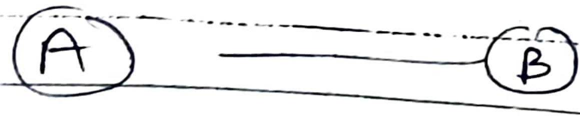
Q. 6 b

1 Complete graph: A graph is said to be completely connected if every node is connected to every node.

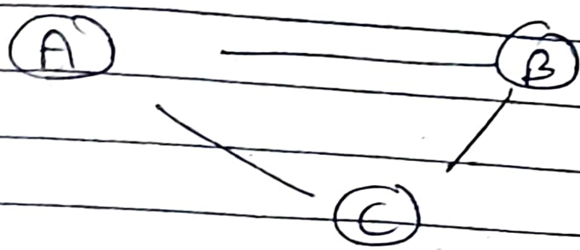
A ~~graph~~ complete graph with 'V' vertices has $\frac{V*(V-1)}{2}$ edges.

Ans:-

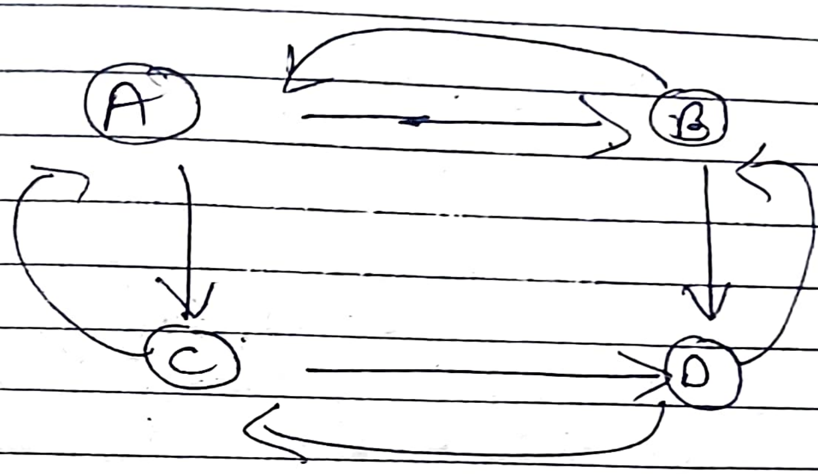
(1)



(2)



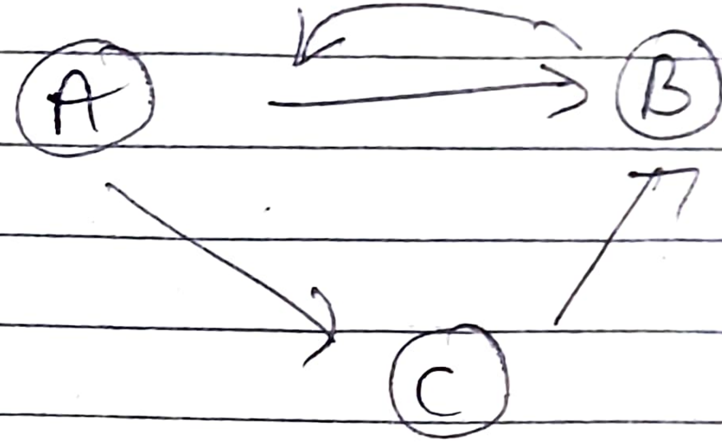
(3)

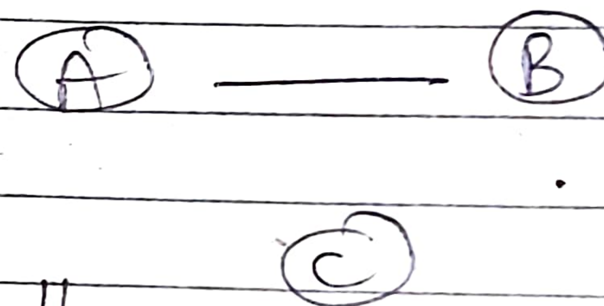


(2) Strongly Connected graph:-

A graph in which every node is reachable to every node. If there is a path from u to v then surely there will be a path from v to u.

Eg:

1 

2 

⇓

It is not a strongly connected graph.

Course code: CST 252

Roll no: CSU20W-3066

Rukhsana

Q.1)(a)

```
void delete ( struct Array* arr[], int n){
    for(int i=0; i<n; i++){
        for(int j= i+1 ;j<n ;j++) {
            if ( ~~ST~~ Array.arr[i] ==
                         Array.arr[j]);
                ~~del ( Array. arr, j, n)~~
                del (Array.arr, j, n);
            }
        }
    }
}
```

Course Code :- CST 252 Page no:- 30/33
Roll no :- CSU2060-3006
Rekhamgari

```
void del ( struct Array* arr, int j, int n)
{
    if ( j == n-1) {
        n = n-1;
    } else {
        while ( j < n) {
            Array.arr[j] = Array.arr[j+1];
            j++;
        }
}
```

Course Code :- CST252

Roll no :- CSU20W-3006

Rekharani

Q.4) (1)

→
```
struct emp{
    int id;
    int salary;
    char name[100];
};

void print(struct emp e[], int id){
  for(int i=0; i<10; i++){
    ~~if(emp emp.e[i]==i~~
    if(emp.e[i].id == id){
      printf("Name: %s",
              emp.e[i].
              name);
      printf("~~Roll no~~ Salary: %d",
              emp.e[i].salary);
```

Page no: 32/33

Course code: CST252
Roll no: CSU20W-3006
Rakhangari

}

}

}

```
int main ( ) {
    struct emp e[10];
    // 10 employees;
    int id = 5;
    print( emp, id);
    return 0;
}
```