Name- Sajidul Islam Saief

ID :- 2020-1-60-149

Ans to the Q.No: 2

ⓐ

② I don't think reversing a circular Linked List is awkward.

```
void reverse()
{ Node* head = NULL;
  Node* current= head;
  Node* *prev=NULL, *next=NULL;
  while(current != NULL) {
    next = current -> next;
    current -> next = prev;
    prev = current;
    current = next;
  }
  head = prev;
}
void push (int data)
{ Node *temp = new Node(data);
```

```
        temp -> next = head ;
            head = temp;

    }
}
```

So reversing of a circular Linked List is not awkward. because a circular Liked List is where all nodes are connected from a circle.

Ans to the Q.No: 4

(4)

```
int main(void)
{
 vector<int> numbers;
 int previous_number[5];
 while (1)
 {
   int tmp;
   cin >>tmp;
 number.push_back (tmp);
```

```
if (tmp<0)
{
    int count=0;
    int pos_mums,_Less_than_5=1;
 for( int i=0 numbers.size()-2; i>=0;i--) {
  if (numbers[i] > 0) {
     if (numbers (i) <5){
     pos_nums_less_than_5 =1;
       break; }
previous_numbers[count]= numbers[i];
          count++;
     }
   if (count == 5)
         break;
    }}
if (count <5 || pos_num_Less_than_5 ==1)
    { cout<<"Display Nothing" <<end;
                                    break;
}
else
{ count<<endl;
for(int i=5; i>=0;i--)
cout<<previous_numbers[i]<<endl;
 cout <<endl
  }
 }
}
```

Ans to the Q.No:6

(a) (i)

```
int main( )
{ int num=3;
 for (int i=0 ; i<2; i++){
   int data
   cin>> "~~data is~~ data;
```

(d)

```
new code = new~~@~~ Node(data);
    if(head == NULL){
      head = newnode;}

else{
    Node *tem= head, *tem1 = temp;

   while (~~tem->~~ != NULL)

  { temp1 = temp;
  ~~t~~temp= tem = tem->next;
  } tem1 ~~=~~-> next = newnode;
  }
  ~~void newnode~~
   newinsert (3);
}
```

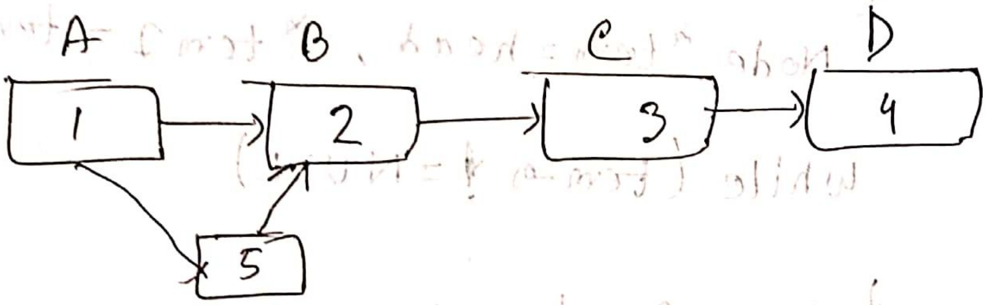
(b)

```
{
          Same Like (i)

          new insert (1);
                                }
```

(b)

No each of the input do not work properly. If I want to insert at any posit it is not work.



```
void insertanypos (int da)
{ Node * newn= new Node(da);
   int pos=2;
  Node *temp = head, *temp 2;
  for (int i=1 ; i< pos; i++)
  { temp 2 = temp;
    temp = temp -> next;
  }
  temp 2 -> next = newn;
     newn -> next = temp;
}
```

Ans to the Q:No: 5

(a)

```
Class Node
{ public:
  int ID;
  double mark;
  Node *prv;
  Node *next;
 Node(int iD)
 {
   ID = iD;
   prv = NULL;
   next = NULL;
 };
}
```

(b)

```
void insenatbeg (double man)

{ Node *newn= new Node(man);
 newn -> next = head;
 head -> prv= newn;
  head =newn;
}

int main( )

{
 int num = 3;
 for(int i=0;i<num;i++)

{
 int ~~data;~~ iD;
cout << "Enter the data"<<endl;
cin >> ~~data~~ iD;
new node = new Node (~~data~~ iD) ;
if (head = NULL)
```

```
        head = newnode;

else{
    Node *temp = head , *temp1 = tem;

    while(tem!=NULL)
    {
     temp1 = temp;
     tem= temp -> next;
     }
    temp1 -> next = newnode;
    newnode -> prv= tem1;
    }
    insentbag (10);
    insentbag(20);
    insentbag (30);
}
```

Ans to Q. No: 2

we need to use queue to solve this problem.

when a customer comes in the Line, he needs to wait for his term. This is very much similar to first in first out (FIFO) so queue is the best condition in this case.

=>

```
int f=0;
Count<<"Counter 2:";
for (int i=0; i<10 ;i++){
if( queueMain→show()>=18&& queuemain→
                            Show()<30)
{ cout<<queuemain →show()<<" ";
  f=1; }
```

```
{
    temp = queueMain -> show();
    queueMain -> deque();
    queueMain -> enque(temp);
      f = 1;
  }
  if (f==0) cout << "NULL" <endl;
  else {
      cout <<endl;
      f =0; }
  cout << "Counter=: ";
  for(int i=0; i<10 ;i++)
  { if ( queuemain -> show () >= 35 && queueMain ->
    show () < 30) {
      cout << queueMain -> show() <"  ";
        f =1; }
    temp = queuemain -> show ();
```

```
        queue Main → deque();
      queueMain → enque(temp);
        ~~flag~~ f = 1;
   }
    if( f == 0) cout<< "NULL"<endl;
   else{
        cout<<endl;
         f=0; }
   cout <<"counter 3 : ";
 {if (queue main→ show()<< " ";
       ~~flag~~ f = 1 }
   temp = queue Main → show();
    queuemain → enqueue(temp);
        (f=1;
 } if ( f== 0) cout << "NULL"<endl;
   else { cout<<endl;
          ~~f=0;~~
       }
```

Ans to the Q. No: 3

(a) To mointoin ~~the~~ a propen onden, the best idea to use queue. In queue we can limit the number of students, Each time a student annives and sit on a chain then othens have to wait. It is similian as FIFO. So queue is the best choice to solve this

(b)
```
int isfull (Queue *queue)
{ neturn (queue-> size = queue->capacity)
}
void enque (Queue *queue, int i)
{ if ( isFull (queue))
     neturn;
queue -> rear( queue -> rear+1)

        queue => capacity;

queue -> array[queue->rear] = i;

queue -> size = queue -> size+1;

cout << i << " enqueue to queue<<
                              endl;

}
```