PAGE
DATE

Lecture

Revision

Question 1

inorder static linear list.

insert function?

our sol

```
bool list::insert(const dataElemType &e)
  { if (noofitems >= maxsize list)
        return false;
    else{ if (noofitems == 0)
        { List[0] = e; noofitems++; }
        else { for (i=0; i< noofitems; i++) {
                 if (List array[i] >= e)
                   for (j=noofitems-1; j <= i; j--)
                     { List[j+1] = list[j]; }
                 break; }
                 list[i] = e;
                 noofitems++;
                 return true;
    }
```

زى ما قلنا فى المحاضرة.

لكن الغوما هنا انا منزود فى الـ iterations

وكده كده الى قد بعض هيبقوا ورا بعض

بس فى الحقيقة هو trade off

علشان بال = انا بعمل Shift

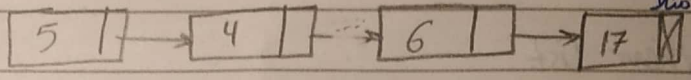
فى الـ array أقل.

ممكن كمان حل .. انك تعمل insert لكل الـ elements وفى الآخر ارتبهم. بس ده طويل أوى و ده أسوء حل.

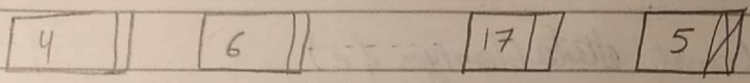
والدكتور عجبه الحل ده وعدل عليه الحاجات الى بالأسود

Question2,

عندنا list كده مثلاً



وعايزين نعمل rotation لـ left كده



our soln

```
List::Rd(void)
{ ElemType & Temp;
  First (Temp);
  while (!(current->next))
    { current->item = current->next->item;
      current = current->next;
    }
  current->item = temp
  return;
}
```

في مشكلة اننا نسينا الـ Special case و الـ empty loop
وتاني مشكلة اني غيرت في الـ current
اللي هستخدمه في function تانية
فكده هيبقى بعد الـ while دي اتغيرت

Dr's solution:

```
List::RoL(Void)
{ Link Ptr;      ListElemType temp;
  ptr = head;
  temp = head->elem;
  if (ptr == NULL)
      return;
  while (ptr->next != NULL) {
      ptr->item = ptr->item->next;
      ptr = ptr->next;
  }
  ptr->item = temp;
}
```

تقريبا دي الدكتور اتلخبط فيها وكانت الغوض ← تبقى item زي باقي الـ list

Another solution (without while loop)

```
List::Rol(void)
{ assert(!emptylist())
  Link temp;
  tail -> next = head;
  temp = head -> next;
  temp
  head -> next = null;
  tail = head;
  head = temp;
}
```

another better method

فكرتها اني بدل ما احرك الـ List
اخذ الـ head اعمله بعد الـ tail
كده يبقى كأني عملت shift left

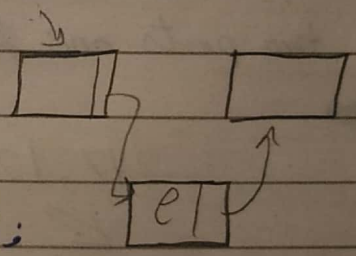
```
tail -> next = head
tail = tail -> next;
head = head -> next;
tail -> next = Null;
```

Question 3

Insert after (const ListElemType &e, LinkP)
insert a node containing e after, node pointed to by P.

our sol

```
Void insertafter(const listtype &e, LinkP)
{ link addedNode;
  addednode = Newnode;
  addednode -> item = e;
  if (P == tail)
    added node -> next = NULL;
  else added node -> Next = P -> next;
  P -> next = added Node;
}
```



الدكتور قال اننا مش محتاجين نعمل الـ Condition بتاع الـ tail
وكمان محتاجين نعمل assert وفي الآخر نرجع المكان الـ heap

the rest of Question 3

Insert Before (const ...

No loop is allowed.

Sol

```
insert before (const ListElemType &e, Link P){
    insertafter (const ... )
    ListElemtype temp;
    temp = P->next->ilem;
    P-> next->item = P->item;
    P-> item = temp;
    }
```

لو في الا متحانه الدكتور قال define الـ class .. اه بتكتب الـ class بالتعريفات بتاعت الـ variables و الـ functions .. لكن لو مقالش بستخدم الـ variables اللي احنا متعودين عليها بس كل طول.

Question 4:

List * FrontBackSplit : member function of List splits List into two parts one for the first half & the second for me.

مثلاً

{17, 12, 5, 4, 23}

{17, 12, 5} {4, 23}

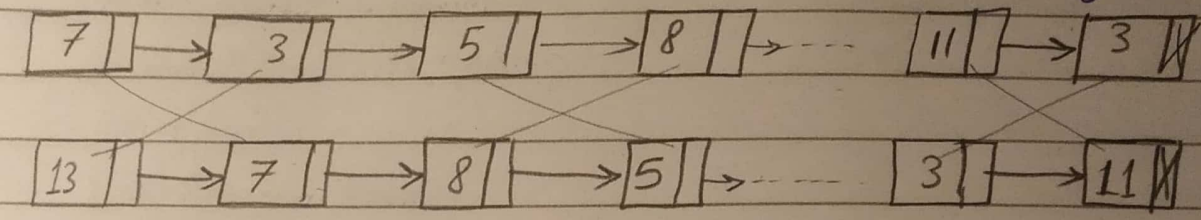
Old List (this) ← returned by function

خد بالك انه اللي بيرجع هنا هو pointer من نوع list
في الأول بنعد عشان اعرف هقف عند
انهي item و split عنده

```
List* List :: FrontBacksplit () {
      List* part2 = new list;
      Assert (part2 != NULL);
      int i=0;
      Link temp = head;
      While (temp != NULL) {
            i++;
            temp = temp -> next;
      }
      if (i%2 != 0) i++;        ⟹ if (i%2 == 0) i++;
      i = i/2; temp = head;      ← دي هنعملها في الحالتين
      for (int j=0; j<i; j++)
            temp = temp -> next;
      Link temptail = tail;
      This.tail = temp;   tail -> next = NULL;   ← نقفل الـ list بقى
      part2 -> head = temp -> next;
      part2 -> tail = temptail;
      return part2;
}
```

Question 5

member function interchange. (loops are valid & it's needed here by the way)



pointer = pointer -> next -> next

pointer -> next بس لازم الأول يبقى عندي