Lecture 08

Lecture contents:

-Inorder list definition &requirements.
-Inorder list operations.
  -Insert function.(general case & special cases).
-Pseudo code
-Inorder list implementation
-List traversal

المرة دى هناخد الinorder list اللى ممكن نرتبها سواء ابجدى او بارقام او او..

Inorder list definition &requirements:

Definition:
an inorder list is a list that is kept (maintained) in some order.
ال order دة ممكن يبقى ascending لو عايزين الترتيب تصاعدي او descending لو تنازلي
أما ال linked list اللى كنا بناخدها قبل كده يطلق عليها اسم bag عشان مش لازم لها ترتيب معين
An inorder list needs two requirements:
1.some part of the stored information must be designated as key.
الkey ده هو الحاجة اللى هنحددها اللى يتعمل على اساسها الترتيب .. حاجة بتميز كل item عن التانى مثلا كل واحد فى السكشن بيميزه الbench number .. يبقى هو ده الkey.
2. a rule for ordering keys.
i.e. for two keys K1 &K2, we must be able to evaluate K1<=K2.
يعنى لازم الkey اللى اختاره ده اعرف اعمل عليه مقارنة او expression يقولى true or false ..زى مثلا لما نقارن بين الbench no بتاعك وبتاع واحد تانى ...فاه نقدر نعمل الexpression بتاع الاكبر من او يساوى ده ويطلع true او false يبقى الkey ده تمام.
ناخد مثال صغير نشوف منه الkey ممكن يكون ايه؟

```
struct student{
        int student_id;
        string name;
        int section;
        int benchno;
        int grades[12];
}
```

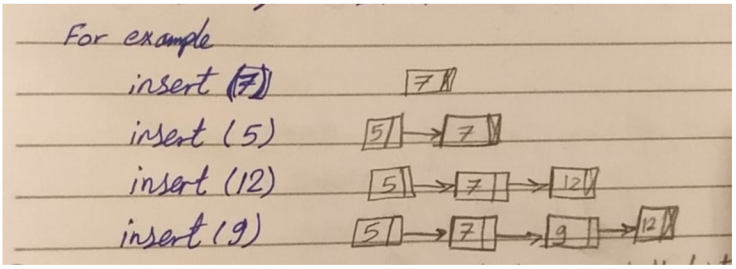
فهنا ممكن الkey يبقى الname
باننا نرتب ترتيب ابجدى .. طب افرض كان فى adham و ahmed ؟
هنقارن كده(adham<=ahmed) فده كده هيبقى true اذا adham قبل ahmed .
وممكن اقارن طول الاسماء زى مثلاahmed وali فهنا ali هييجى الأول.. المهم اننا قدرنا هنا نحقق الشرطين نجيب key يميز كل item عن التانى والkey ده نقدر نعمل expression يقارن ويدى true او false.
طيب
ممكن كمان يبقى الkey هو الbench number ..يبقى ترتيب بالارقام.وممكن يبقى الname والbenchno مع بعض ... وممكن يبقى بالgrades وهكذا.

Inorder list operations:

insert – first – next
عادى .. اصل ايه اللى هيختلف؟.. وممكن معاهم delete اللى المفروض نعملها لوحدنا من الlist اللى فاتت

كل الoperations زى اللى كتبناه قبل كده بالضبط .. حاجة واحدة بس هى اللى الimplementation بتاعها هيتغير..الinsert .
از اى؟
كده..

هتلاقى ان الinsert مش من مكان معين زى زمان .. اما كان من الtail بس .. لا فى تانى سطر الinsert كان من ناحية ال head وتالت سطر ال12 جت من الtail ورابع سطر ال9 جت لا من الhead وللا من الtail من النص كده فى مكانها

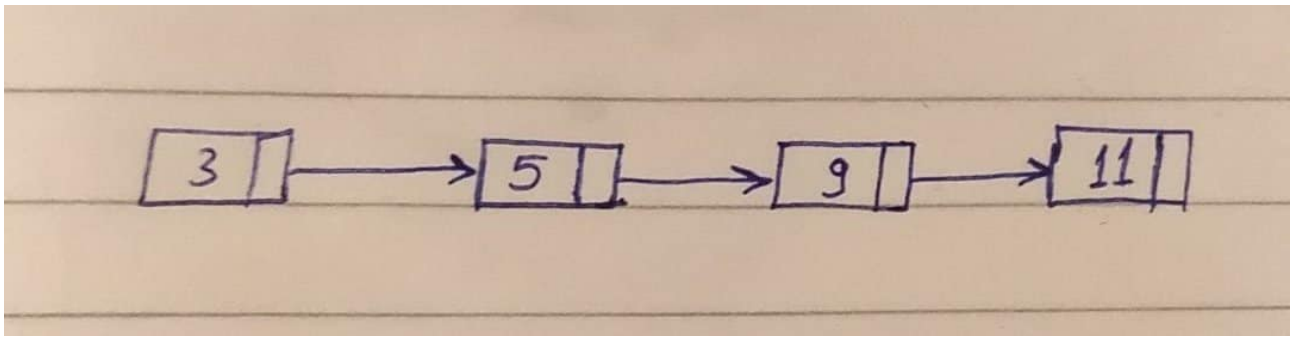


المضبوط بالترتيب.
هيبقى يللا ندرس الinsert.

Insert function:

فى special cases وفى general case خلينا ناخد الgeneral الأول عشان هنعرف معاها كام تعريف جديد.

General case:



لو هعمل :

```
insert(7);
```

المفروض اعرف بالضبط تتحط فين .. فهفضل اسأل هل ال3 اكبر من ال7 ...لا.. طب هل ال5 اكبر من ال7 .. لأ .. طب هل ال9 اكبر من ال7 .. اه .. يبقى ال7 هى الnode اللى قبل ال9 وبعد ال5.
طب افرض كان بدل ال9 دى 7؟؟
كنت هحط السبعة اللى معايا قبل اللى فى الlist عادى .. وهتبقى الinorder list عادى .. وعشان كده الدكتور قال خد بالك السؤال ميبقاش هل ال5 اكبر من ال7 ... لا المفروض من الاول يبقى السؤال هل ال5 اكبر من او تساوى ال7؟؟
متنساش او تساوى دى .. مادام ال keys اللي في ال list مش unique (زي الرقم القومي مثلا .. مفيش اتنين ليهم نفس الرقم القومي وبالتالي مستحيل نلاقي ال operator '=')
المهم فى المثال اللى معانا ده وصلنا ال7 هتبقى بين ال5 وال9 .. هنسمى الnode اللى الpointer بتاعها هيشاور على ال node الجديدة (pred(predecessor node
واللى الnode الجديدة الpointer بتاعها هيشاور عليها او يعنى اللى بعد الpred فى الlist .. يبقى اسمها pred→next
يعنى لو عايز ال 5 هقول

```
pred→item=5;
```

ولو عايز ال7 هقول

```
addednode→item=7;
```

ولو عايز ال9 هقول

```
pred→next→item=9;
```

وكأنى مدخلة للnext اللى بيشاور على الpred عشان يجيب هو الnext بتاعه وبعد كده هات بقى الitem اللى فيه.
فكده expression المقارنة هيكون كده

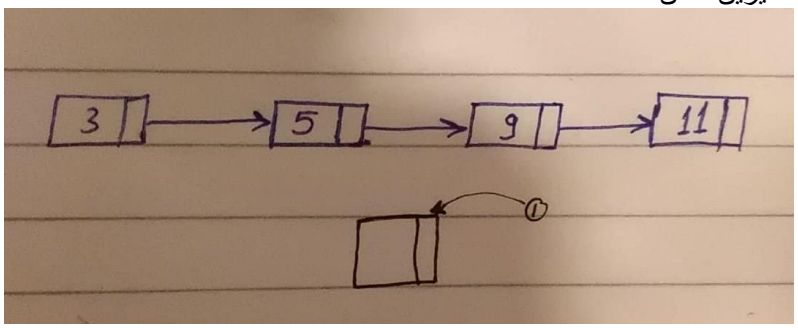
```
pred→item<addednode→item && addednode→item <= pred→ next → item
```

كده انا بتأكد ان الرقم اللى هزوده اكبر من اللى قبله و اقل او يساوى اللى بعده .. بس الشرط ده مش valid لو ال pre→next→item=NULL يعنى لو على المثال اللى عندى ده هقول
insert(17) ..فاللى بعد ال addednode ولا حاجة اللى هو ال NULL ال condition ده مش هيطلع true بسبب الجزء التانى اللى بعد ال&& مع ان ال20 المفروض تتحط فى الاخر فعلا.. فدى special case هنحلها بان لو ال loop خلصت ال condition فضل false يبقى هى دى حالة ان الرقم اللى هنحطه هو اخر رقم فى ال list .. هنشوف هنشوف :"D

طيب ..نرجع للى كنا فيه.. خطوة خطوة كده هنحط ال7 ازاى.

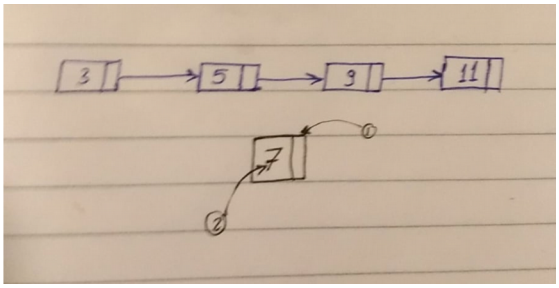
1.allocate

يعنى نقول لل heap اننا عايزين مكان

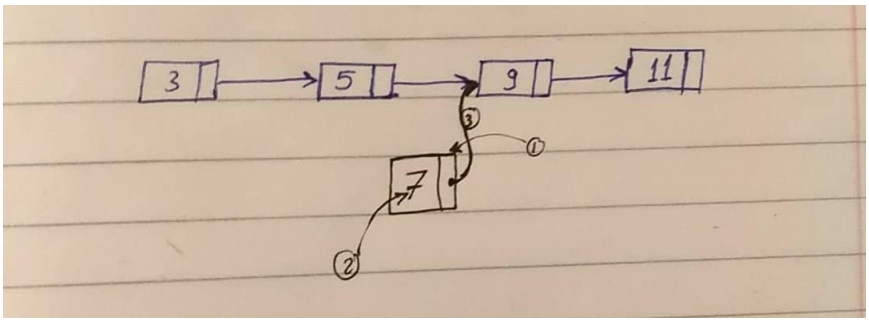


2.store item.

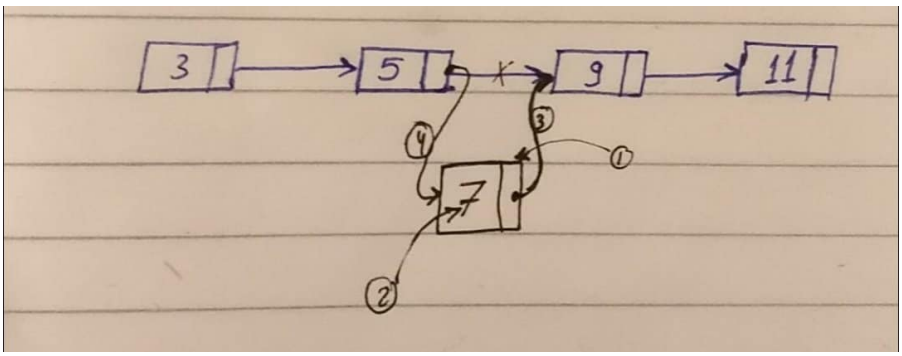
نخزن ال7 فى المكان اللى خدناه



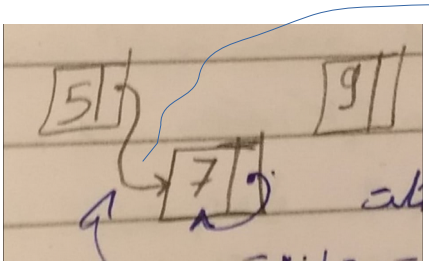
3.هخلى ال node دى ال pointer بتاعها يشاور على ال node اللى فيها ال9



4.هخلى ال node اللى فيها ال5 ال pointer بتاعها يشاور على node ال7 الجديدة.



هل لو الخطوة 3 بدلت مع الخطوة 4 هيحصل حاجة؟
ولا اى حاجة ..كارثة بس :D
لانك هتخلى الpointer بتاع ال5 يشاور على الnode الجديدة.. لغاية هنا تمام .. كمل معايا بقى .. الpointer بتاع ال node الجديدة ده هتخليه ازاى يشاور على node ال9 ؟؟
الpointer اللى كان بيشاور على ال9 اللى كنت هاخد قيمته خلاص راح خليته يشاور عليا (الnode الجديدة يعنى) .. فهاجى اشاور على الnext→pred هيطلع انا وابقى بشاور على نفسى لانى فى الخطوة دى بال logic بتاعنا غيرت ال next→pred لnode ال7.(اللى هو "تصدق سلخت قبل ما ادبح" :'D)



كده فهمنا ايه اللى هيتعمل فى الgeneral case ..الدكتور بيقول ان الكلام ده ممكن يتعمل بarray وان ده exercise بيجى فى الامتحانات.

طيب بالنسبة للspecial cases بقى

<u>special cases:</u>

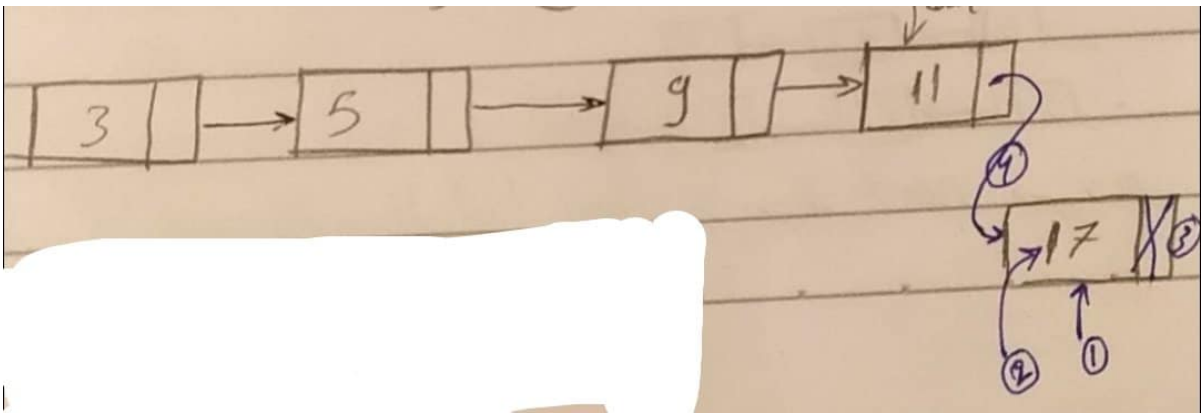
1.empty list:

لو بعمل insert والlist لسة فاضية ..
1. ناخد من الheap مكان .. allocate.
2. نخزن القيمة فى المكان.
3.هنخلى الpointer يشاور على null
4. هنعرّف ان الnode دى هى الhead.

ومش هنحتاج ال tail هنا (لو فاكر في المحاضرة اللي فاتت الدكتور ادانا exercise مش محتاجين فيه ال tail برضه) .. بس ليه بقى مش محتاجينه؟؟
لو فاكر احنا كنا بنستخدم ال tail عشان نقدر نـ insert نود جديدة بعد اخر node .. بس احنا دلوقتي في ال inorder list مش محتاجين ال option دة فمش هنحطه

2.the “addedNode” item is greater than all of the list items.

دى الحالة اللى قلنا مشكلتها فوق فى الcondition



نفس الخطوات بالضبط ماعدا رقم 3 انى هخلى الpointer يشاور على الNULL

3.the “addedNode” item is less than all of the list items

في ال case دي هنعمل insert في اول ال list خالص .. مثلا لو قولنا (1) insert :
يبقى في الاول هنـ allocate space from heap وبعد كدة هنـ store e .. بعد كدة نخلي ال "next” بتاع ال node الجديدة "اللي اسمها addedNode” يشاور على ال "head” اللي كان موجود قبل ما نعمل insert
وبعد كدة هنخلي ال node الجديدة دي هي ال head بتاع ال list وشكراً

طيب كدة دي ال cases اللي عندنا .. تفتكر نقدر نعمل optimization بحيث إننا مانضطرش نكتب ال 4 cases في الكود (نلاقي طريقة مثلا تخلينا نكتب 3 cases او 2 cases ودة احسن طبعا) ؟؟
تعالى نفكر كدة …

كدة كدة اول خطوتين في كل ال cases واحدة (احنا دايماً بنـ allocate وبعدين نـ store) .. فهنفكر في الخطوتين اللي بعد كدة .. لو فكرنا في special case 1 & special case 3 هنلاقي ان الخطوة التالتة في special case 1 ال next بتاع "addedNode” بيشاور على NULL .. بس في special case 3 هنلاقيه بيشاور على "head”
بس ثواني .. في special case 1، مش كدة كدة اصلا ال "head” معمول له initialization بـ NULL بسبب ال constructor ؟؟ .. يعني في الحقيقة خطوة 3 دي هي هي في special case 1 & special case 3 (لو خليت ال next في الحالتين يشاور على head)
وخطوة 4 في الحالتين بنخلي ال "head” يشاور على addedNode
يعني من الآخر يا معلم ممكن نعتبر ان special case 1 & 3 دول نفس ال case .. مصلحة

طب تعالى نبص برضه على special case 2& general case … اول خطوتين نفس البصمجة اللي بنعملها دايماً .. الخطوة التالتة في ال general case المفروض اننا بنخلي "next” بتاع "addedNode” يشاور على ال node اللي بعد ال predecessor … وفي special case 2 بنخلي "next” بتاع "addedNode” بـ NULL … طب لو انا بدل ما اخليه يشاور على NULL خليته يشاور على ال node اللي بعد ال predecessor برضه، كدة انا بوظت حاجة؟؟ .. الاجابة لأ لأن كدة كدة ال node اللي بعد ال predecessor هتكون NULL فانا برضه كدة بشاور على NULL
اشطة ظبطنا الخطوة التالتة .. تعالى نبص على الرابعة .. هي هي في الحالتين : ال next بتاع ال predecessor بيشاور على ال node الجديدة
يبقى كدة برضه ال general case هي هي special case 2

التفكير اللي فكرنا فيه دلوقتي دة خلانا نقدر نكتب في الكود 2 cases بس بدل ما كنا هنكتب 4 cases وطبعاً دة احسن كتير في ال execution time

طيب تعالى بقى نعمل حاجة جديدة تخلينا نرتب افكارنا اكتر وتسهل علينا كتابة الكود .. هنكتب **pseudo code** .. لو ماتعرفوش ف دة كود وهمي مالوش اي قواعد ممكن نكتبه بال english عادي ولو عايز تكتبه بفرانكو مش مشكلة وبالتالي مالوش constraints او keywords فنقدر نستخدم ال pseudo code واحنا بنكتب بأي programming language .. بنستخدمه عشان نرتب افكارنا قبل ما نكتب الكود .. تعالى نشوف هنكتبه ازاي:

Pseudo code for “insert” function:

If list is empty **or** item <= head → item
==> insert at beginning
else

انا كدة لقيت اني هعمل if condition في الكود .. وهـ check على 2 conditions بـ or operator

/*عايز اروح على المكان اللي هـ insert فيه .. طب هعرفه منين؟؟ هدور عليه يعني هعمل search .. على ال items اللي في ال list دي item by item وافضل اعمل next على ال items دي لحد ما اوصل للآخر .. لو وصلت لل item يبقى هعمل insert لل item الجديدة في المكان اللي لقيته .. ولو ماوصلتش وال list خلصت يبقى هـ insert في اخر مكان خالص .. يبقى واضح اني هحتاج اعمل loop بس قبل ال loop محتاج اخلي ال predecessor يبقى هو ال head عشان اعرف امشي في ال loop : */

pred = head

while (pred → next != NULL && pred → next → item < added item) {

pred = pred → next … ana kda bshoof element by element
}

lw 5aragt mn el loop yb2a 7sl fail ll condition bta3y w hab2a f el mkan elly ana 3ayz a3ml insert fee :

==> insert between pred & pred → next

كدة خلصنا ال pseudo code وعملنا ديزاين وكل حاجة .. نكتب الكود بقى:

هنيجي نكتب ال .h ونسمي الفايل بتاعنا بـ InorderList.h .. هنلاقي اننا بنكتب نفس اللي كنا بنكتبه في list.h بتاع المحاضرة اللي فاتت .. ماعدا بس اننا هنا مش هنحط ال tail عشان مش محتاجينه (((مع اني كان ممكن احطها برضه عادي .. مثلا انا عايز أ check انا وصلت للآخر ولا لأ ودة عن طريق ال tail .. فلو عايز احطها عادي مفيش مشكلة طبعا))) المهم يعني مش هنركز في ال .h وهندخل على ال implementation على طول :

InorderList.cpp

```
bool InorderList ::insert (const ListElemType &e) {
link addedNode;
link pred;

addedNode = new Node; //step 1
if (addedNode == NULL)
        return false;
else {
addedNode→ item = e; //step 2

//step 3
if (head == NULL || e <= head → elem) { //check if empty list or entered number is less
                                        than all items
```

/* هنا بقى فيه مشكلة خطيرة جدا جدا جدا .. دلوقتي دي if condition .. جينا نـ check على head ولقيناها بـ NULL فال condition الاولاني كان true .. لما نيجي نـ check بقى على ال condition التاني، المفروض ان head مش بتشاور على حاجة لأنها NULL بس في نفس الوقت هي مُجبرة انها تبص على حاجة هي بتشاور عليها اسمها elem .. طب

ازاي بقى حاجة مش بتشاور على حاجة وانا بقولها هاتيلي البتاع اللي انتي بتشاوري عليه؟؟؟ظظ ... دة كدة جنان فبيحصل segmentation fault
فبيجي الاستاذ compiler يعمل نفسه ناصح ويقولك خلاص مادام دي or وأول condition كان true فانا مش هـ check على ال condition التاني واروح على الكود على طول .. فل كدة؟؟
لأ مش فل .. لو انا عكست ترتيب ال conditions .. يعني عملت check على ال e الاول قبل ما اعمل check على ال NULL وكان وقتها head == NULL فعلاً فانا كدة برضه رجعت لمشكلة ال segmentation fault
عشان كدة لازم آخد بالي كويس اوي اوي اوي اوي من ترتيب ال conditions .. وعامةً في اي pointer انا المفروض أ check على null دايماً اول حاجة */

```
addedNode → next = head;

//step 4
head = addedNode;
return true;
} // end of if condition

else { //search for proper position
pred = head;
while (pred→ next != NULL && pred→ next → item < e) {
/* order is very dangerous...
                                    /* conditions ال باقي بيكمل مش false حاجة بلاقي compiler ال ما اول .. && ال في

pred = pred → next;
}

//step 3 (2nd case)
addedNode → next = pred → next;

//step 4 (2nd case)
pred → next = addedNode;

return true;
}
```

المشاكل اللي فاتت دي بتقول لي خليك فاكر دايماً إن ال pointers مش بتيجي بالشبه (مش عشان كانت شغالة بالطريقة دي في كود تاني قبل كدة يبقى هتشتغل كمان هنا) .. وبرضه خليك فاكر ان ال order بتاع ال instructions مهم جدا

كدة خلصنا ال inorder list الحمد لله .. بس الدكتور كان نسي يشرح حتة قديمة شوية فهنشرحها دلوقتي ...
دلوقتي لو انا كتبت في كود ال main حاجة زي كدة :

```
main (){
List mylist;
int i,j;

for (int i=0; i<10; i++){
cin >> j;
mylist.insert(j);
}
```

كدة انا عملت insert ل 10 elements .. اشطة؟؟
جيت بقى بعد كدة قلت انا عايز اعمل retrieve ل element معين من ال list .. بس قبل ما اعمل retrieve انا عايز اعمل على ال element دة كام operation كدة ... هنعمل كدة ازاي؟؟

هو دة اللي اسمه List traversal :

List traversal :

بتمشى في ال list واعمل عليها some operations .. بعمل كدة ازاي؟؟
في كود ال main برضه هكتب كدة :

```
int item;
bool flag;
flag = mylist.first(item);

while (flag) {
//do some operations with “item”

//if we want to retrieve the remaining items:
flag = mylist.next(item);
}
```