insert function

```
void insert(btreenode*& sr, int num) {
    if (sr == NULL) {
        sr = new btreenode;
        sr->data = num;
        sr->leftchild = NULL;
        sr->rightchild = NULL;
    }
    else {
        if (num < sr->data)
            insert(sr->leftchild, num);
        else
            insert(sr->rightchild, num);
    }
}
```

عمل الفنكشن:
1. نبدأ من العقدة الجذر (Root)
2. إذا كانت الشجرة فارغة، يتم إنشاء عقدة جديدة وتصبح هي الجذر
3. إذا لم تكن الشجرة فارغة، نقارن القيمة الجديدة مع قيمة العقدة الحالية
4. إذا كانت القيمة الجديدة أصغر، ننتقل إلى الفرع الأيسر
5. إذا كانت القيمة الجديدة أكبر، ننتقل إلى الفرع الأيمن
6. نكرر الخطوات السابقة باستخدام الاستدعاء الذاتي (Recursion)
7. عند الوصول إلى مؤشر فارغ (NULL)، يتم إنشاء عقدة جديدة وتوضع القيمة فيها

Traversing Binary Search Tree

```
// Inorder Traversal (Left - Root - Right)
void inorder(btreenode* sr) {
    if (sr != NULL) {
        inorder(sr->leftchild);
        cout << sr->data << "\t";
        inorder(sr->rightchild);
    }
}

// Preorder Traversal (Root - Left - Right)
void preorder(btreenode* sr) {
    if (sr != NULL) {
        cout << sr->data << "\t";
        preorder(sr->leftchild);
        preorder(sr->rightchild);
    }
}

// Postorder Traversal (Left - Right - Root)
void postorder(btreenode* sr) {
    if (sr != NULL) {
        postorder(sr->leftchild);
        postorder(sr->rightchild);
        cout << sr->data << "\t";
    }
}
```

أنواع Traversal:

Inorder Traversal (Left - Root - Right)
- يتم زيارة الفرع الأيسر ثم العقدة ثم الفرع الأيمن.

Preorder Traversal (Root - Left - Right)
- يتم زيارة العقدة أولًا ثم الفرع الأيسر ثم الفرع الأيمن.

Postorder Traversal (Left - Right - Root)
- يتم زيارة الفرع الأيسر ثم الفرع الأيمن ثم العقدة.

Search function

```
bool search(btreenode* sr, int num)
{
    if (sr == NULL)
        return false;

    if (sr->data == num)
        return true;

    if (num < sr->data)
        return search(sr->leftchild, num);
    else
        return search(sr->rightchild, num);
}
```

عمل الفنكشن:
1. نبدأ من العقدة الحالية (غالبًا الجذر).
2. إذا كانت العقدة فارغة، فهذا يعني أن القيمة غير موجودة في الشجرة.
3. إذا كانت قيمة العقدة الحالية تساوي القيمة المطلوبة، يتم إرجاع true.
4. إذا كانت القيمة المطلوبة أصغر من قيمة العقدة الحالية،ننتقل للبحث في الفرع الأيسر.
5. إذا كانت القيمة المطلوبة أكبر من قيمة العقدة الحالية،ننتقل للبحث في الفرع الأيمن.
6. نكرر الخطوات السابقة باستخدام الاستدعاء الذاتي حتى نجد القيمة أو نصل إلى عقدة فارغة

---

delete function

```
void deleteBST(btreenode* &root, int num)
{
    if (root == NULL)
        return;

    if (num < root->data)
        deleteBST(root->leftchild, num);

    else if (num > root->data)
        deleteBST(root->rightchild, num);

    else
    {
// الحالة 1: بدون أبناء //
        if (root->leftchild == NULL && root->rightchild == NULL)
        {
            delete root;
            root = NULL;
        }
// الحالة 2: ابن يمين فقط //
        else if (root->leftchild == NULL)
        {
            btreenode* temp = root;
            root = root->rightchild;
            delete temp;
        }
// الحالة 3: ابن يسار فقط //
        else if (root->rightchild == NULL)
        {
            btreenode* temp = root;
            root = root->leftchild;
            delete temp;
        }
// الحالة 4: له ابنين //
        else
        {
            btreenode* temp = root->rightchild;
            while (temp->leftchild != NULL)
                temp = temp->leftchild;

            root->data = temp->data;
            deleteBST(root->rightchild, temp->data);
        }
    }
}
```

عمل الفنكشن:
•1 إذا كانت العقدة الحالية فارغة (NULL)
فهذا يعني أن القيمة غير موجودة، وينتهي الفنكشن.

•2 إذا كانت القيمة المطلوب حذفها أصغر من قيمة العقدة الحالية
ننتقل إلى الفرع الأيسر باستخدام الاستدعاء الذاتي.

•3 إذا كانت القيمة أكبر من قيمة العقدة الحالية
ننتقل إلى الفرع الأيمن باستخدام الاستدعاء الذاتي.

•4 عند العثور على العقدة المطلوبة يتم الحذف حسب حالتها:

- الحالة الأولى: العقدة بدون أبناء
يتم حذف العقدة مباشرة وجعل المؤشر NULL.

- الحالة الثانية: العقدة لها ابن أيمن فقط
يتم استبدال العقدة بابنها الأيمن ثم حذف العقدة القديمة.

- الحالة الثالثة: العقدة لها ابن أيسر فقط
يتم استبدال العقدة بابنها الأيسر ثم حذف العقدة القديمة.

- الحالة الرابعة: العقدة لها ابنان
يتم إيجاد أصغر قيمة في الفرع الأيمن،
نسخ هذه القيمة إلى العقدة الحالية،
ثم حذف العقدة التي تحتوي على تلك القيمة.