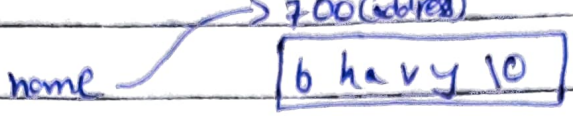
Copy Constructor

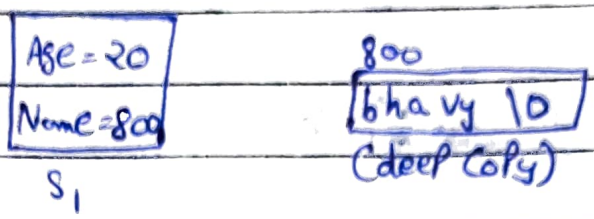
Code →

```
Class Student {
    int age;
    Public:                          name is "Public"
    char * name;
    Student (int age, char *name) {
        this -> age = age
        // Shallow Copy
        // this -> name = name;

        // Deep Copy                    { we are using deep
        this->name = new char[strlen(name) +1];   Copy as can be seen
        strcpy (this -> name, name);
    }
    Void display(){
        ____________;
    }
};
int main() {
    char name [ ] = "bhavy";
    Student s1(20, name);
    s1.display;
⟹  Student s2(S1);

    s2.name[0] = 'x';
    s2. display();
    s1.display();
}
```

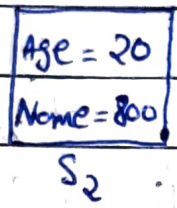
output :→ bhavy 20
xhavy 20 ← changed
xhavy 20 ← S1.display();

→ Lets See why such happen ⟶





⟶ how, we are making S_2 with help of Copy Constructor; & what it does is Copy the values as it is from S_1 to S_2 or it make the Shallow Copy



& Now, both S_1 & S_2 Pointing towards Same Array

& Hence, changing only in S_2 will be Easily get Reflected to our S_1.

Sol^n → we need to Customize our Copy Constructor so as it makes deep Copy

Lets modify Copy Constructor —

→ In our class only —

```
Student (Student S) {
        this → age = S.age;
        this → name = S.name;
}
```

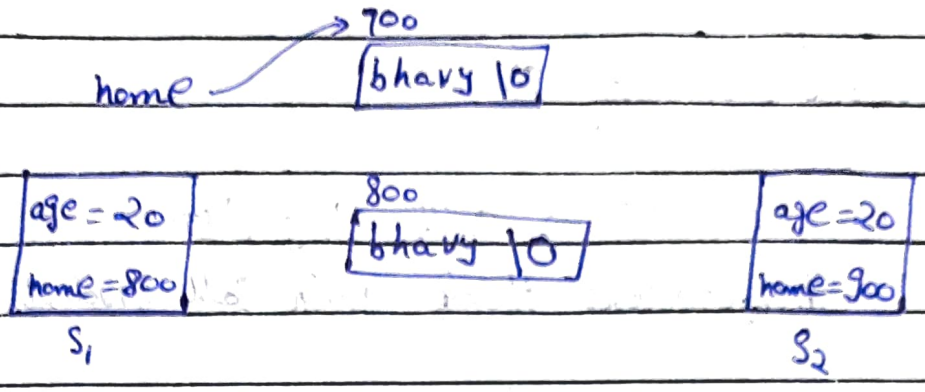
{ this is inbuilt Copy Cont; & creats Shallow Copy }

⟶ for Deep Copy —

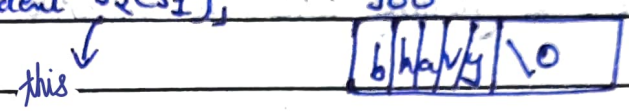
```
Student (Student S) {
        this → age = S.age;
        this → name = new [Strlen(S.name)+1];
        Strcpy (this → name, S.name);
}
```

{ deep Copy Constructor }

This time ——



Student S2(S1);



⇒ There is Still one issue, Lets See into it ——

Student (Student S) Student S2(S1)

i.e, Student S = Main S1 ;

→ Main mai Jo S1 banaya usko Pass kara Rahe what we ultimately telling to do is
Copy Values of S1 from Main to Student S
i.e, we are calling again Copy Constructor

Student S2 = S1 ; → Copy Co- call Hota hai
Student S = Main.S1 → " " " " ——.

→ & we know as we now created our own Copy Constructor default is Vanished.

So, the Problem exist is

Student (Student S)

Student S calls Copy Constructor
then, Student itself again calls "Student S"

⟶ i.e, It is getting into an infinite loop Calling Copy Constructors only.

Sol^n what we want is on calling Student S, we don't call Copy Constructor again
i.e, Pass object as Pass by Reference is a Possible Sol^n

Recap of Pass by Refer

int i = 5; ⟶ [5] i

int j = i; ⟶ [5] j

int &j = i;
" no j is created as a new block"

[5] j

i.e, [5] i, j

In Case of function what happen is ⟶

Pass by Value

```
int fun(int j) {
    int j = Main.i;
}
```

[5] j

main

```
int i = 5
fun(i)
```

```
int fun(int &j) {
    int &j = Main.i
}
```

(No new block, j will Point to i in main)

So, the Simple Solution is —

Student (Student &S)

⇒ but now, we need to avoid changes

i.e, int i = 5;

int &j = i;

i++j ⟶ 6

j++j ⟶ 7

but we want no chages for j i.e, j++ ✗

⇒ for that we need to do ⟶

int Const &j = i ; (it is Constant now)

So,

Student (Student Const &S)

i.e, now we Can Read values of "S" but cannot make any illegal ⇒ changes in it.

output :— bhavy 20
xhavy 20
bhavy 20