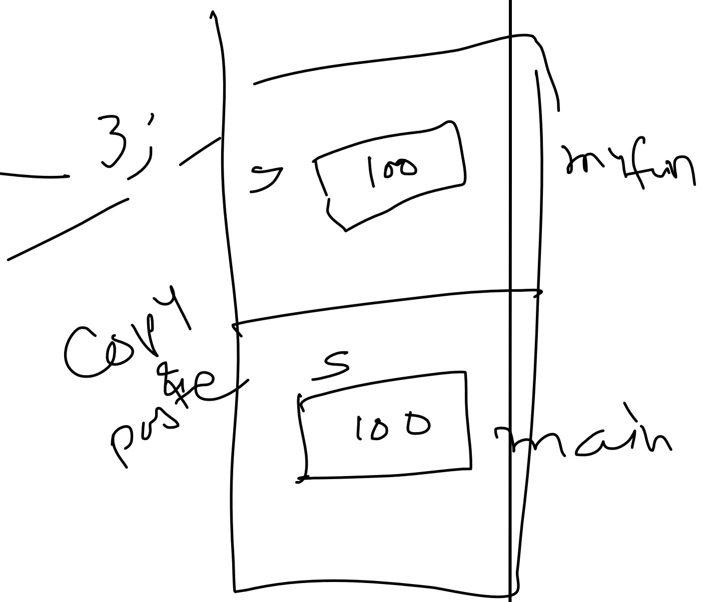
Let’s solve it

By default,
Structure variable is passed by value when used as actual

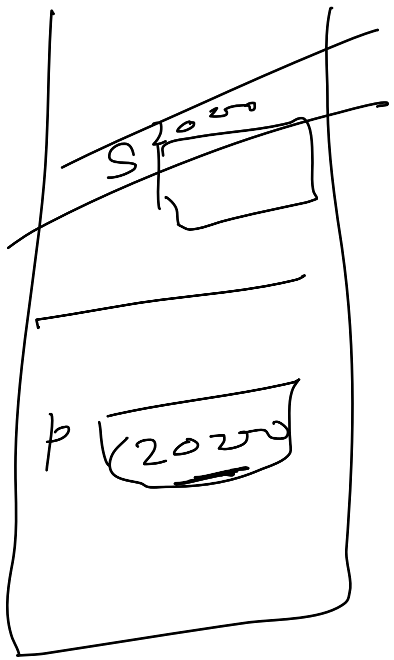
```
int main()
{
    struct student s = { ______ };
    myfun(s);
}

void myfun(struct ____ s)
{
    s.id++;
}
```



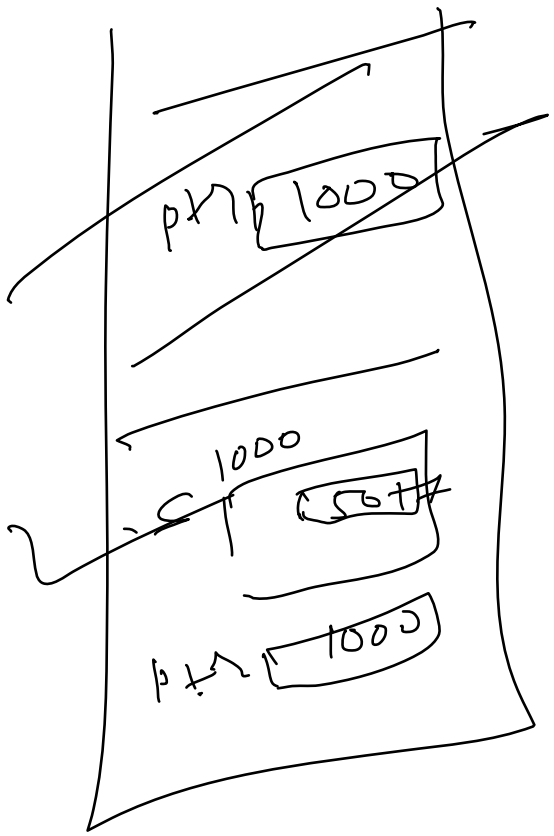
```
int main()
{
    struct student s = {...};
    myfun(&s);
    printf("%d", s.marks); // updated.
}
void myfun(struct student *ps)
{
    (ps->marks)++; // (*ps).marks)++;
}
```

Syntactically no errors. Logically error bcoz
local variable s gets destroyed

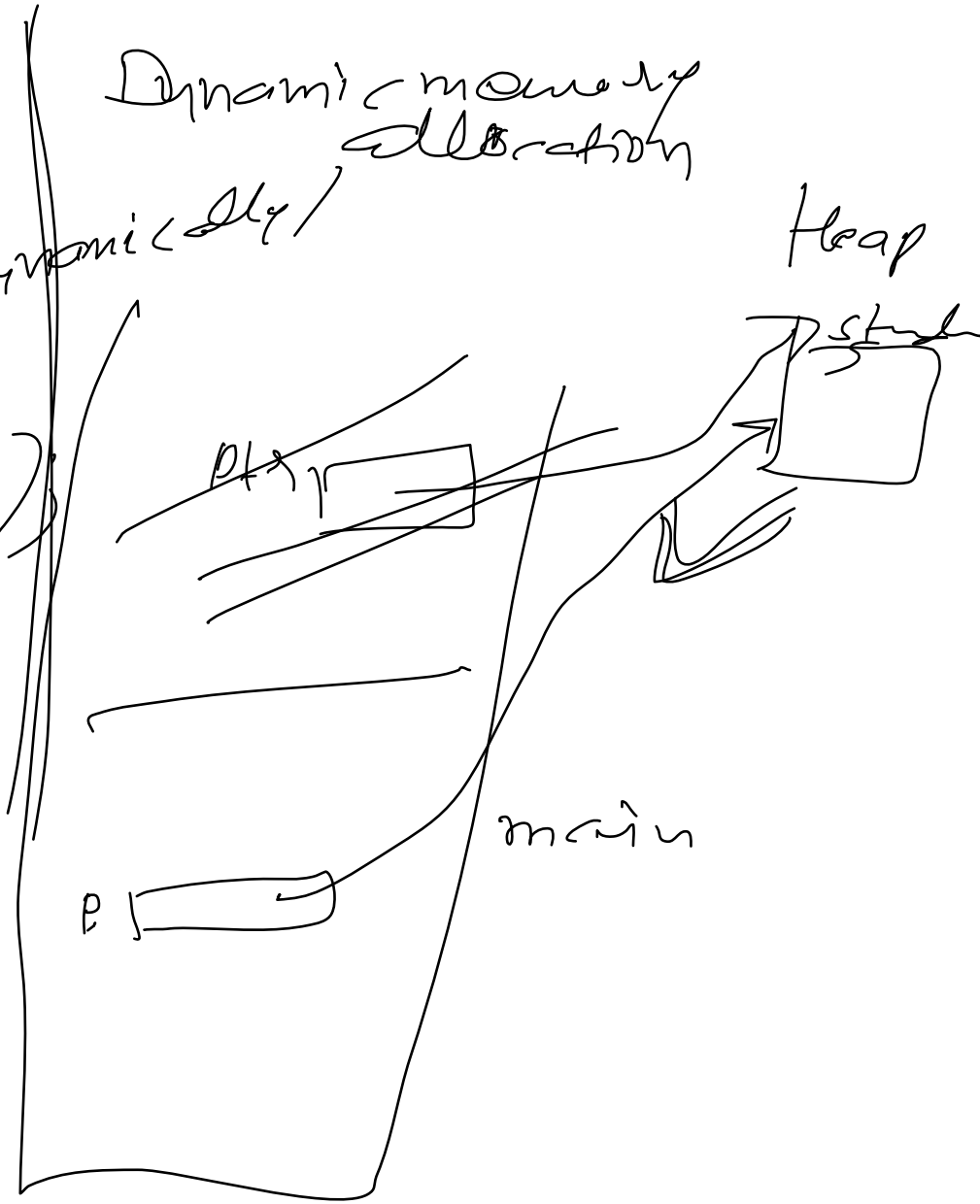
```
struct student * myfun2(        )
{
    st—— stude  s = { —— };      // local variable
    struct student *ptr = &s;

    return( &s );
    //or return( ptr );
}
main()
{ struct student *p;
    p = myfun2(        )
    p——("%d", p->marks);
}
```



```
struct student * myfun3 ( struct student *ptr)
{
    ptr->marks++;
    return(ptr);
}
int main()
{ struct student s = {____};
  struct student *ptr;
  ptr = myfun3( &s );
  printf("%d", ptr->marks);
}
```



```
struct student *myfunc()
{
    struct student *ptr;
    ptr = (st_stud *) // allocate memory dynamically
          malloc ( sizeof (st_student));
    return ptr;
}
main()
{ st_student *p;
  p = myfunc();
  p-> ("%d", p->marks);
  free ( p ); // Deallocate memory
}
```



ptr = &s;

s.name , s.id . . —

(*ptr) float x = 20.32;

(*ptr).name = x;

(*ptr).name

(*ptr).id

ptr->name - - -

struct NAME
{
int count;
float *p;
} *ptr, s;

(*ptr).p

*ptr->p ~ *(ptr->p)

~

sm = *ptr->p++;

(*(ptr->p))++.

Arrow has higher priority

5000++ ?

20.32++ ?

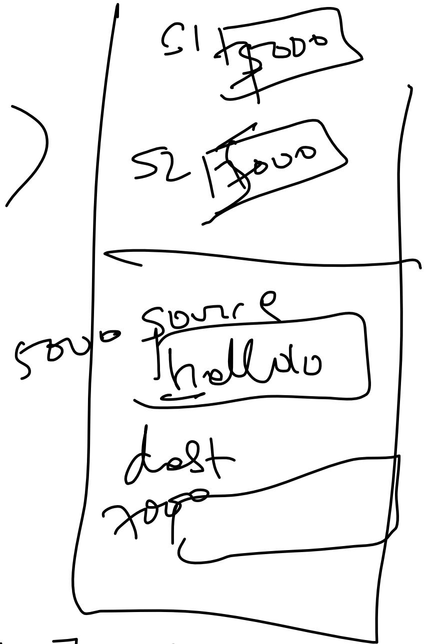
1000++ !

formal args — S2, s1 are pointers only *, [ ].

```
char* strcopy ( char *s1 , char *s2 )      // x = y = 2 +10  (10)
                     dest          source
{
    while( (*s1++ = *s2++ ) != '\0');
} return
```

data itself

```
strcopy ( char s1[ ], char s2[ ] )
{  while ( 1 )
   {  if ( *s2 == NULL)
          break;
      *s1 = *s2;
      s1++;   s2++;
}
strcopy( dest, source);
```



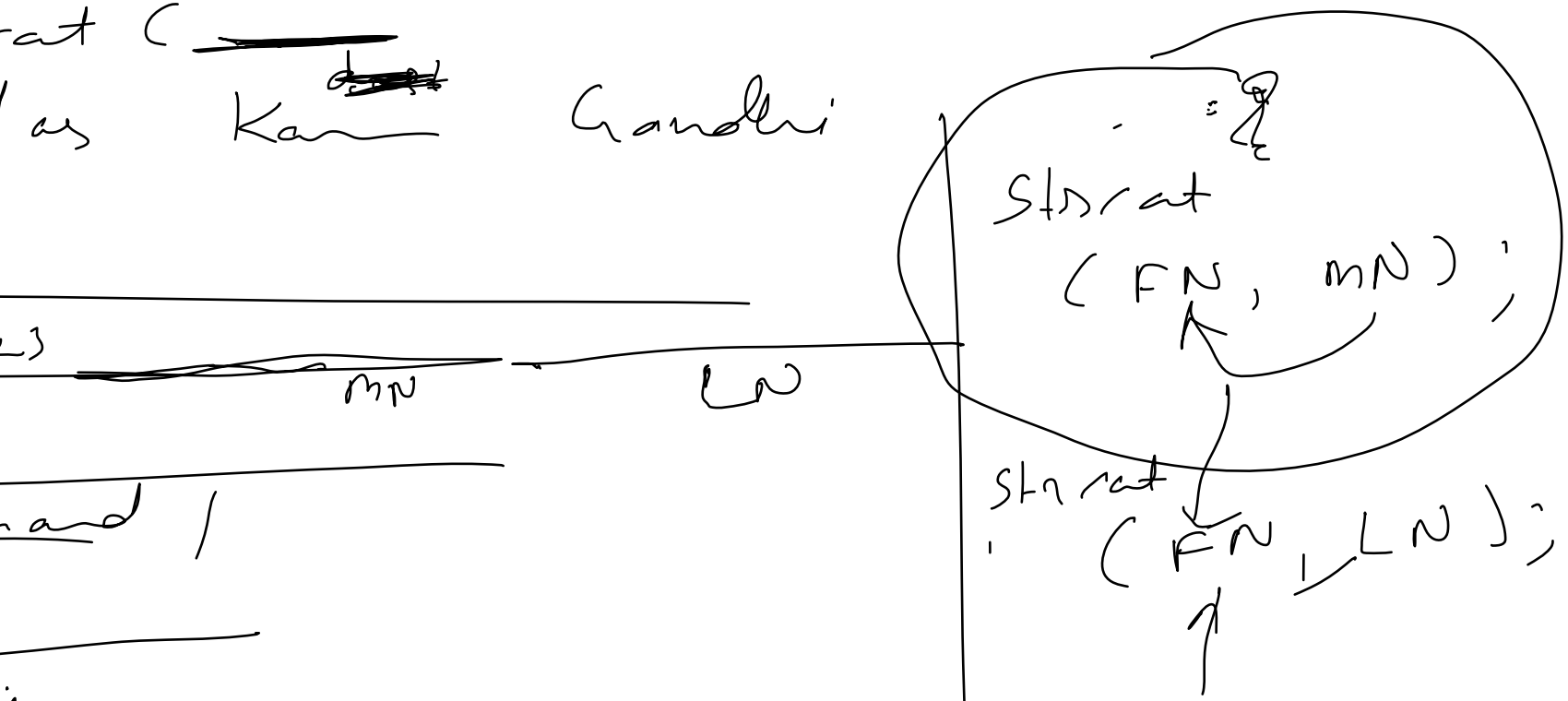

```
char sour[50], dest[50];
"hello";
```

7000

Addresses of

char* Strcat (

Mohandas Karamchand Gandhi



FN Mohandas

mN Karamchand

LN Gandhi

Strcat ((strcat(FN, mN), LN);