Question 2: (30 points) In this question, you will use the following Student class definition.

```cpp
#include<iostream>
using namespace std;

class Student {
public:
    Student( int num = 0 ){
        id = num;
        cout << "Constructor for " << id << endl;
    }
    Student( const Student &S ){
        id = S.id;
        cout << "Copy constructor for " << id << endl;
    }
    ~Student(){
        cout << "Destructor for " << id << endl;
    }
    void operator=( const Student &S ){
        id = S.id;
        cout << "Assignment operator for " << id << endl;
    }
    void setId( int num ){
        id = num;
    }

private:
    int id;
};
```

Write the output of the following programs (you can ignore memory leaks, if any). Write your answers in between the given outputs (you will also need to use the space after the last given output). The space in between these outputs may not reflect the number of lines to be displayed.
(a) (8 points)

```cpp
void bar(int a, int b){
    Student P(a);
    static Student R(b);
    if (a < b)
        Student S(a + b);
    cout << "after if--" << endl;
}
int main(){
    cout << "before first call---" << endl;
    bar(40, 10);
    cout << "after first call---" << endl;
    bar(20, 70);
    cout << "after second call---" << endl;
    return 0;
}
```

Answer:
before first call---
Constructor for 40
Constructor for 10
after if---
Destructor for 40
after first call---
Constructor for 20
Constructor for 90
Destructor for 90
after if---
Destructor for 20
after second call---
Destructor for 10

1 point each, but you may lose some points if you have extra lines.

You may get zero from here, if you have extra lines.

- **A1** Destructor for 10 (extra line) \(-1\) point
  *Destructor are not called for static objects when you exit the function. They are called at the end of the program.*

- **A2** If you have any call for 70 (extra line) \(-1\) point
  *such as Constructor for 70, Copy Constr. for 70, Assignment op for 70*
  *Static objects are created just once when you first call the function.*

- **A3** If you have this line after the \(\rightarrow 0\) point
  *This object's lifetime ends with the if block. Thus, its destructor is called when you reach the end of the if statement.*

- **A4** If you have Destructor for 70 at the end (here) extra line
  \(-1\) point
void foo(Student X, Student &Y, Student *Z)
{
    X.setId(10);
    Y.setId(20);
    Z->setId(30);
    Y = X;
    Z = &Y;
}

int main()
{
    Student A(5), B(6), C(7);
    // The outputs before this part has been given
    // DO NOT rewrite them; they will NOT be graded
    cout << "before function call--" << endl;
    foo(A, B, &C);
    cout << "after function call--" << endl;
    return 0;
}

Answer:
Constructor for 5
Constructor for 6
Constructor for 7
Before function call---
Copy constructor for 5
Assignment operator for 10
Destructor for 10
After function call---
Destructor for 30
Destructor for 10
Destructor for 5

1 point each, but you may lose some points if you have extra lines

B1 If you have two lines for assignment
    Assignment operator for ... ? - 1 point
* Z = &Y statement does not call the assignment operator since they are
  pointers (they are not objects)

B2 If you have more extra lines for
    extra constructor, extra copy constructor,
    extra assignment operator for the
    parameters ... - 1 point
* foo(Student X, Student &Y, Student *Z)
  calls the copy constructor does not call anything since it is pass-by-
  anything since it is pointer reference parameter

B3 for each extra line - 1 point

B4 More than one destructor call - 1 point
* The destructor is called
  only for the parameter X
  since Y is a pass-by-reference
  parameter and Z is a pointer

For any order
(c) (8 points) Ignore memory leaks, if any.

```c
void bar(Student A, Student *B){
    B[1].setId(77);
    B++;
    A.setId(55);
    B[1] = A;
    B = new Student(66);
}
int main(){
    Student *Q = new Student[4];
    Q[0].setId(10);
    Q[1].setId(20);
    Q[2].setId(30);
    Q[3].setId(40);
    // The outputs before this part has been given
    // DO NOT rewrite them; they will NOT be graded
    cout << "before bar---" << endl;
    bar(Q[0], Q + 1);
    cout << "after bar---" << endl;
    delete []Q;
    return 0;
}
```

Answer:

<table>
<thead>
<tr>
<th>Constructor for 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constructor for 0</td>
</tr>
<tr>
<td>Constructor for 0</td>
</tr>
<tr>
<td>Constructor for 0</td>
</tr>
</tbody>
</table>

Before bar---

Copy constructor for 10
Assignment operator for 55
Constructor for 66
Destructor for 55

After bar---

Destructor for 55
Destructor for 77
Destructor for 20
Destructor for 10

1 point each, but you may lose some points if you have extra lines.

---

C1: If you have a Destructor for 66 in between before/after or on an extra line after the after bar--- statement → -1 point

Examples are:

- CC 10
- A 55 or D 55 or D55
- D 20 or D 10 or D10
- B 66

C2: Copy constructor for 20 or Assignment for 20 (or any place in between before bar and after bar)

* bar (Student A, Student *B)

this is a pointer (not an object). Thus, it does not call the copy constructor or the assignment

C3: If you have an extra line for 77 such as

Assignment for 77
Copy constructor for 77
Destructor for 77

* setId function changes the data member of an object, it does not create the object. Thus, it does not call any constr/destr/copy constr/assignment

* This destructor was called if you have delete B statement in the bar function
(d) (8 points) Ignore memory leaks, if any.

```c
void foo(Student A, Student *B, Student **C){
    B[1] = &A;
    *C = B;
    A.setId(66);
    C[1][0] = A;
}
int main(){
    Student **Y;
    cout << "before---" << endl;
    Y = new Student *[2];
    int count = 11;
    cout << "after---" << endl;
    for (int i = 0; i < 2; i++){
        Y[i] = new Student [2];
        for (int j = 0; j < 2; j++){
            Y[i][j].setId(count);
            count++;
        }
        cout << "after i = " << i << "---" << endl;
    }
    cout << "before foo---" << endl;
    foo(Y[0][0], Y[1], Y);
    cout << "after foo---" << endl;
    delete []Y[0];
    delete []]Y;
    return 0;
}
```

Answer:

before---
after---
Constructor for 0
Constructor for 0
after i = 0---
Constructor for 0
Constructor for 0
after i = 1---
before foo---
Copy constructor for 11
Assignment operator for 11
Assignment operator for 66
Destructor for 66
after foo---
Destructor for 11
Destructor for 66

You may lose some points, if you have extra lines.

Any output = -1 point
Y = new Student *[2]; it creates an array of object pointers, not an array of objects. Thus constructors are not called

if you have one line or more than two lines, you lose all points (no partial)

if you have too many incorrect lines, you may get zero or lose some points (depending on your extra lines)

4 lines or more = 0 point
3 lines (but 2 are incorrect) = 0 point