Define a class called Student with ID (string), name (string), units (int) and gpa (double).

Define a constructor with default values to initialize strings to NULL, unit to 0 and gpa to 0.0.

Define a copy constructor to initialize class members to those of another Student passed to it as parameters.

Overload the assignment operator (=) to assign one Student to another.

Define a member function called read() for reading Student data from the user and a function called print() for printing Student data to screen. These functions will need to work correctly for both Student objects and Student\_worker objects as defined below.

Derive a class called Student\_worker from Student class with the additional data of weekly\_hours (int) and hourly\_wage (double).

Define for Student\_worker the same constructors and overloading functions as for Student. The constructors must also initialize the additional data of weekly\_hours and hourly\_wage to 0 and 0.0, respectively. The read and print functions must also read and print the additional data of weekly\_hours and hourly\_wage.

In main, declare an array of 100 Student pointers and have the program keep creating either a Student or Student\_worker, asking each time if the Student being added is a student worker (see the example below), dynamically allocating memory for each, storing the pointer returned by new in the array of Student pointers and reading data for reach by calling the corresponding (the correct) read() function, until no more is desired by answering with a 'n' or 'N' to the question: "Enter a student?".

Display all Students and Student\_workers created and read by calling the correct print() function (belonging to Student or Studentn\_worker, as the case may be for each).

The following is an example of interaction between the user and the program:

Enter a student? [y/n]: y

Is this a student worker? [y/n]: n

Enter ID, name, units and GPA:

Alice Gordon 101 24 3.3

Enter a student? [y/n]: y

Is this a student worker? [y/n]: n

Enter ID, name, units and GPA:

Robert Palmer 102 18 3.1

Enter a student? [y/n]: y

Is this a student worker? [y/n]: y

Enter ID, name, units and GPA:

Brian Gomez 103 24 3.5

Enter weekly hours: 25

Enter hourly wages: $11.50

Enter a student? [y/n]: y

Is this a student worker? [y/n]: n

Enter ID, name, units and GPA:

Cindy Martinez 104 12 2.9

Enter a student? [y/n]: y

Is this a student worker? [y/n]: n

Enter ID, name, units and GPA:

Jo Smith 105 21 3.7

Enter a student? [y/n]: y

Is this a student worker? [y/n]: y

Enter ID, name, units and GPA:

Mary Mullins 106 18 3.0

Enter weekly hours: 20

Enter hourly wages: $12.50

Enter a student? [y/n]: n

You created the following students:

Alice Gordon, ID: 101, units: 24, GPA: 3.3

Robert Palmer, ID: 102, units: 18, GPA: 3.1

Brian Gomez, ID: 103, units: 24, GPA: 3.5

Weekly hours: 25, hourly wages: $11.50

Cindy Martinez, ID: 104, units: 12, GPA: 2.9

Jo, Smith, ID: 105, units: 21, GPA: 3.7

Mary Mullins, ID: 106, units: 18, GPA: 3.0

Weekly hours: 20, hourly wages: $12.50

Press any key to continue.

Note that student workers have the additional data of weekly hours and hourly wage read and printed. To get full credit, you must have one array of Student pointers and store the pointer for all Student objects - Students and Student\_workers - in that one array and your program must read and print the required data for each depending on whether it's a Student or Student\_worker and you must provide all required data and member functions.

You do not need to provide separate files but if you do, please, do not zip them. After uploading, press submit. You should see the indication that it's been submitted.