Name: ______________________

NetID / Email: ____________________
1. The following program is supposed to compute “overtime pay” for an employee, but it isn’t working correctly. If a user works 40 or fewer hours in a week they are entitled to “straight pay” (their hourly rate multiplied by the hours worked). If they work more than 40 hours they are entitled to “time and a half” on the extra hours (i.e. 50 hours per week at $10 per hour would yield $10 per hour on the first 40 hours and $15 per hour on the 10 “overtime” hours”)

Rewrite / annotate the program so that it functions correctly.

```python
rate = float(input("How much do you make per hour? "))
hours = float(input("How many hours did you work this week? "))
if hours <= 40:
    pay = rate * hours
else:
    reg_pay = rate * 40
    ot_pay = (hours-40) * (rate*1.5)
    pay = reg_pay + ot_pay
print(“Your total pay is”, pay)
```

Rewritten program:

```python
rate = float(input("How much do you make per hour? "))
hours = float(input("How many hours did you work this week? "))
if hours < 40:
    pay = rate * hours
else:
    reg_pay = rate * 40
    ot_pay = (hours-40) * (rate*1.5)
    pay = reg_pay + ot_pay
print("Your total pay is", pay)
```
2. Trace the output of the following short programs:

<table>
<thead>
<tr>
<th>Program</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 6</td>
<td>A</td>
</tr>
<tr>
<td>y = 12</td>
<td>B</td>
</tr>
<tr>
<td>z = 18</td>
<td>D</td>
</tr>
</tbody>
</table>
| if z % x == 0:  
  print ("A")  
if y % x == 0:  
  print ("B")  
if x + y != z:  
  print ("C")  
else:  
  print ("D") |
| x = 20  | A      |
| y = 46  | B      |
| z = 28  | C      |
| if x + z > y:  
  print ("A")  
elif x + y > z:  
  print ("B")  
elif y + z > x:  
  print ("C")  
else:  
  print ("D") |
| x = 10  | A      |
| y = 20  | B      |
| z = 30  | C      |
| if x < y:  
  if x + y == z:  
    print ("A")  
  else:  
    print ("B")  
elif y < z:  
  if z - y == x:  
    print ("C")  
  else:  
    print ("D")  
else:  
  print ("E") |
3. Write a program that simulates the concept combinational padlock. Assume the code to open the lock is 23 – 17 – 8. Remember, padlocks require the user to input their code in the following manner:

1. Turn right to 23
2. Turn left to 17
3. Turn right to 8

Also remember that a combinational lock **does not** inform the user that they made an error until they enter all three numbers. If the user makes an error, the program should prompt the user to try again after all three values are entered. Here’s a sample run of the program:

```
while True:
    n1 = int(input(“Enter number 1: “))
    n2 = int(input(“Enter number 2: “))
    n3 = int(input(“Enter number 3: “))

    if n1 == 23 and n2 == 17 and n3 == 8:
        print (“Lock opened with combination 23-17-8”)  
        break
    else:
        print (“Invalid code, try again”)  
```

```
Enter number 1:  23
Enter number 2:  17
Enter number 3:  29
Invalid Code! Try again!

Enter number 1:  23
Enter number 2:  17
Enter number 3:  8
Invalid Code! Try again!

Enter number 1:  23
Enter number 2:  17
Enter number 3:  8
Lock opened with combination: 23-17-8
```
4. Write a “calculator” program that asks the user for two numbers as well as an “operation code” (“a” for add, “s” for subtract, “d” for divide or “m” for multiply). Using the information provided perform the specified operation and print the result. Here is a sample running of the program:

```
Number 1: 2.0
Number 2: 3.0
Operation (a/s/d/m): a
Invalid operation! Try again.
Operation (a/s/d/m): d
2.0 + 3.0 = 5.0
```

Note that you cannot assume that the user will enter a valid operation code (i.e. they could type in the string “multiply” instead of the string “m”). In this case you will need to present the user with some kind of error (i.e. “Sorry, that’s not a valid operation code”) and re-prompt them. However, you can assume that the user will input valid floating-point numbers when prompted.

Also note that dividing a number by 0 will result in a runtime error. Prevent this from happening in your program by providing special output in this case (i.e. 5.0 / 0.0 = undefined)

```python
number1 = int(input("Number 1: "))
number2 = int(input("Number 2: "))
while True:
    code = input("Operation (a/s/d/m): "))
    if code == "a" or code == "s" or code == "d" or code == "m":
        break
    else:
        print ("Invalid code, try again")

if code == "a":
    print (number1, "+", number2, "=", number1 + number2)
elif code == "s":
    print (number1, "-", number2, "=", number1 - number2)
elif code == "m":
    print (number1, "*", number2, "=", number1 * number2)
else:
    if number2 == 0:
        print (number1, "/", number2, "= undefined")
    else:
        print (number1, "/", number2, "=", number1 / number2)
```
A small college has asked you to write a program for their admissions department to help them determine if a student should be accepted into their school. Write a program that uses the following criteria to determine whether a given applicant should be admitted:

- Combined SAT score of 1600 or more
- A high school GPA of 3.0 or higher
- At least 3 extracurricular activities

However, this particular school places a heavy emphasis on extracurricular activities, so students with 5 or more activities only need a 1400 combined score on their SAT and a GPA of 2.8. Comment your code as necessary. You can assume that the user will enter floating-point values.

Here is a sample running of your program. Note that you should ask the user if they want to repeat the process for additional students when you are finished.

Student name: Craig
Combined SAT Score: 1800
High school GPA: 3.2
# of extracurricular activities: 3
Craig should be admitted!
Another student? yes

Student name: John
Combined SAT Score: 1500
High school GPA: 3.1
# of extracurricular activities: 7
John should be admitted!
Another student? yes

Student name: Chris
Combined SAT Score: 1300
High school GPA: 2.9
# of extracurricular activities: 8
Chris should not be admitted.
Another student? no
while True:
    name = input("Student name: ")
    sat = int(input("Combined SAT score: "))
    gpa = float(input("High school GPA: "))
    extra = int(input("# of extracurricular activities: "))
    if sat >= 1600 and gpa >= 3.0 and extra >= 3:
        print(name, "should be admitted")
    elif sat >= 1400 and gpa >= 2.8 and extra >= 5:
        print(name, "should be admitted")
    else:
        print(name, "should not be admitted")
again = input("Another student? ")
if again == "no":
    break
6. Write a program that asks the user to enter in an unlimited number of products. Users will supply both the name of a product and its cost. You can assume the user will always supply cost values that are greater than $0.00. The user should be able to continually enter products until they supply choose to stop by typing the string “end”.

Afterward you should print out the total cost of the user’s order, the average cost per item and a listing of all product names that they purchased. Format any currency output to two decimal places. Here’s a sample running of your program:

```
Enter the name of a product, 'end' to end: apple
Enter the price for this product: 0.99

Enter the name of a product, 'end' to end: pear
Enter the price for this product: 1.29

Enter the name of a product, 'end' to end: peach
Enter the price for this product: 1.99

Enter the name of a product, 'end' to end: end

Total cost: $ 4.27
Average cost of each item: $ 1.42
Products purchased: apple pear peach
```
# accums
total = 0.0
num_items = 0
product_names = ""

# get products
while True:
    # get a product name
    name = input("Enter product name, 'end' to end: ")

    # see if they are done here
    if name == 'end':
        break

    # get the price
    price = float(input("Price: "))

    # update accum vars
    total += price
    num_items += 1
    product_names += name + " "

# output
print("Total: ", total)
print("Average: ", total / num_items)
print("Num items: ", num_items)
print("Product names: ", product_names)