Answer the questions in the spaces provided. If you run out of room for an answer, continue on the back of the page.

| Question | Points | Score |
|---|---|---|
| 1 | 8 | |
| 2 | 6 | |
| 3 | 10 | |
| 4 | 10 | |
| 5 | 15 | |
| 6 | 20 | |
| Total: | 69 | |

Name: ______________________________

Section: ______________________________

1. (8 points) Name 4 different primitive types.

2. **For loops and while loops**

   (a) (3 points) When should you use a while loop, and when should you use a for loop?

   (b) (3 points) In Python, you can always replace a single while loop with a single for loop, as they are functionally equivalent if used correctly. **T** **F**

3. Given the following piece of code:

```
x1 = [ ... ]
x2 = [ ... ]

output = []
for item1 in x1:
    i = str(item1)
    for item2 in x2:
        result = i + str(item2)
        if not result in output:
            output.append(result)
print(output)
```

   (a) (7 points) What is printed when `x1 = [1,2,3]` and `x2 = [2,3,4]`?

   (b) (3 points) Describe, in no more than 3 sentences, what this code does.

4. (10 points) Write code that returns the minimum and maximum of a list of integers `S` with only one pass (only one loop should be used). You are *not* allowed to use the functions `min` or `max` in this code (but may do so on other questions). The first line of code is given to you.

```
S = [ ... ]     # the contents of S should be irrelevant
```

5. (15 points) The Fibonacci numbers $F_n$ are defined in the following recursive way:

$$F_n = \begin{cases} F_{n-1} + F_{n-2} & n \geq 2 \\ 1 & n = 1 \\ 0 & n = 0 \end{cases}$$

For example, the first 15 Fibonacci numbers are $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377$.

Write a piece of code that uses iteration to calculate the $73^{\text{rd}}$ Fibonacci number and prints it out.

6. (20 points) **Selection sort** is a sorting algorithm that is not very efficient. For an array (i.e., a list) of $n$ elements, it takes time proportional to $n^2$ in order for selection sort to sort it correctly.

   The way selection sort works is by repeatedly finding the minimum element of a subset of the array and swapping it with the first element in that subset until the array is sorted. For example, consider the following array:

   $$[2, 4, 3, 1, 6, 5]$$

   On the first iteration, selection sort looks at the whole array and finds that 1 is the minimum element, so it swaps 2 and 1, giving us:

   $$[1, 4, 3, 2, 6, 5]$$

   Then, since selection sort knows that the first element is sorted, it restricts itself to the subset containing the elements $[4, 3, 2, 6, 5]$. In this subset, the minimum element is 2, so it switches 4 and 2:

   $$[2, 3, 4, 6, 5]$$

   It continues without switching any more elements for the next two iterations (as 3 and 4 are in their correct places) until it gets to the last two elements, which it swaps:

   $$[5, 6]$$

   Expanding the view again gives us the sorted list:

   $$[1, 2, 3, 4, 5, 6]$$

   Write code that performs selection sort on a list of integers `S`. The first line is written for you.

```
S = [ ... ]     # the contents of S should be irrelevant
```