Do not turn this page until you have received the signal to start.
(Please fill out the identification section above, write your name on the back of the test, and read the instructions below.)

Good Luck!
Question 1. [4 marks]

Part (a) [1 mark] Write the output of the code below in the box below it.

```python
x = 3
y = x - 1
x = 4
print(x, y)
```

Part (b) [1 mark] Write the output of the code below in the box below it.

```python
s = '0123456789'
i = 0
while i < len(s) and int(s[i]) < 2:
    print(s[i])
i = i + 1
```

Part (c) [1 mark] Fill in the box with Python code that will make the program behaviour match the comments. You may not make any other changes to the code or add code outside the box.

```python
def check_carry_on(weight, liquid_free):
    ''' (float, bool) -> bool
    Return True if and only if the bag’s weight is less than 22
    and the bag does not contain liquid as indicated by liquid_free.'''
    return
```

Part (d) [1 mark] Fill in the box with Python code that will make the program behaviour match the comments. You may not make any other changes to the code or add code outside the box.

```python
s1 = "computer"
s2 = "midterm"

# Using only s1, s2, concatenation, and indexing and/or slicing, print the string 'code'.
print(
```

```
Question 2. [6 marks]

Part (a) [4 marks]

For the function below, complete the function header (using a meaningful function name) and write a docstring (including the type contract, description and two examples).

```python
def
    result = ''
    for ch in s:
        if ch.isdigit():
            result = result + ch + ch
        else:
            result = result + ch
    return result
```

Part (b) [2 marks]

Write a Python program that prompts the user to enter a string, calls the function from Part (a) passing the string that the user entered as an argument, and prints the value returned by the function. You may not change the function.
Question 3.  [10 MARKS]

Part (a)  [5 MARKS] Complete the function according to its docstring.

VOWELS = 'aeiou'
HIDDEN = '~'

def points_for_hidden_consonants(puzzle, view, points_by_position):
    ''' (str, str, str) -> int

    Return the points for the hidden consonants for this view of puzzle. Points are awarded based on the position of the hidden consonants. points_by_position is a string of digits that define the number of points awarded for a hidden consonant in the corresponding position of the view. Hidden vowels do not earn points.

    >>> points_for_hidden_consonants('apple', '^^^^e', '31211')
    4
    >>> points_for_hidden_consonants('cat', '^at', '311')
    3
    >>>
    '''
Part (b) [5 marks] A piano teacher says that students in grade 5 and up should practice 10 minutes per song daily, but students in grades 4 and below only need to practice 5 minutes per song. But the teacher also has a daily limit on the total time spent practicing and says to stop when the limit is reached, even if the student has not finished practicing each individual song for the recommended amount of time. Her daily limit is as follows:

- students in grades 1-2: 15 minute limit
- students in grades 3-4: 30 minute limit
- students in grade 5 and up: 60 minute limit

Complete the following function according to the description above and the docstring below.

```python
def practice_needed(num_songs, grade_level):
    ''' (int, int) -> int
    Return the number of minutes of practice needed per day according to this teacher for num_songs at the level grade_level.
    >>> practice_needed(5, 4) # 5 songs at grade 4 level
    25
    >>> practice_needed(4, 2) # 4 songs at grade 2 level
    15
    '''
```

Short Python function/method descriptions:

```python
__builtins__:
    input([prompt]) -> str
    Read a string from standard input. The trailing newline is stripped. The prompt string,
    if given, is printed without a trailing newline before reading.
max(a, b, c, ...) -> value
    With two or more arguments, return the largest argument.
min(a, b, c, ...) -> value
    With two or more arguments, return the smallest argument.
print(value, ..., sep=' ', end='
') --> NoneType
    Prints the values. Optional keyword arguments:
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
```

```python
int:
    int(x) -> int
    Convert a string or number to an integer, if possible. A floating point argument
    will be truncated towards zero.
```

```python
str:
    S.count(sub[, start[, end]]) -> int
    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end]. Optional arguments start and end are
    interpreted as in slice notation.
S.find(sub[,i]) -> int
    Return the lowest index in S (starting at S[i], if i is given) where the
    string sub is found or -1 if sub does not occur in S.
S.isalpha() -> bool
    Return True if and only if all characters in S are alphabetic
    and there is at least one character in S.
S.isdigit() -> bool
    Return True if and only if all characters in S are digits
    and there is at least one character in S.
S.islower() -> bool
    Return True if and only if all cased characters in S are lowercase
    and there is at least one cased character in S.
S.isupper() -> bool
    Return True if and only if all cased characters in S are uppercase
    and there is at least one cased character in S.
S.lower() -> str
    Return a copy of S converted to lowercase.
S.replace(old, new) -> str
    Return a copy of string S with all occurrences of the string old replaced
    with the string new.
S.split([sep]) -> list of str
    Return a list of the words in S, using string sep as the separator and
    any whitespace string if sep is not specified.
S.startswith(prefix) -> bool
    Return True if S starts with the specified prefix and False otherwise.
S.strip() -> str
    Return a copy of S with leading and trailing whitespace removed.
S.upper() -> str
    Return a copy of S converted to uppercase.
```