THE UNIVERSITY OF AUCKLAND

SUMMER SEMESTER, 2016
Campus: City

COMPUTER SCIENCE
TEST

Principles of Programming

(Time Allowed: 75 minutes)

NOTE:
You must answer **all** questions in this test.
No calculators or watches are permitted
Answer in the space provided in this booklet.
There is space at the back for answers which overflow the allotted space.

Surname	
Forenames	
Student ID	
Username	

Q1 (/10)	**Q3** (/10)	**Q5** (/10)
Q2 (/10)	**Q4** (/10)	**TOTAL** (/50)

ID: .....

Question 1 (10 marks)

a) Complete the output produced by the following code:

```
result = 3 * (4 + 1) - 8 // 3 + 3 ** 2
print("Result:", result)
```

```
Result:
```

(2 marks)

b) Complete the output produced by the following code:

```
word1 = "cheers"
word2 = "holiday"
word3 = word1[1] + word2[-1]
pos1 = word1.rfind("ee")
pos2 = word2.find("ad")
word3 = word3 * 3
print("word3:", word3, "pos1:", pos1, "pos2:", pos2)
```

```
word3:              pos1:           pos2:
```

(2 marks)

ID: .....

c) Complete the `get_extended_word()` function which has one string parameter. The function gets a random letter from the parameter string and inserts two copies of the random letter immediately after the random letter (i.e., in total there will be three occurrences of the random letter side by side inside the string). The string which is returned by the function is in uppercase. For example, the following program using the completed `get_extended_word()` function might give the output:

```
From happy to HAPPYYY
From happy to HAAAPPY
From aBC to AAABC
```

```
import random
def get_extended_word(word):
```

(6 marks)

```
def main():
    word = "happy"
    extended_word = get_extended_word(word)
    print("From", word, "to", extended_word)

    word = "happy"
    print("From", word, "to", get_extended_word(word))

    word = "aBC"
    print("From", word, "to", get_extended_word(word))

main()
```

ID: .....

Question 2 (10 marks)

a) Assume that the integer variable, `value`, has been initialised. Write a boolean expression which evaluates to `True` if `value` is an odd number between 16 and 33 both inclusive but not equal to 29:

(2 marks)

b) Give the output produced by the following code:

```
def some_ifs(a, b, c):
      if a > b and not a > c:
          print("A")
      elif not (a > b or a == c):
          print("B")
      else:
          print("C")

          if a > b or a > c:
              print("D")
          elif not(a > b or a > c):
              print("E")
      print("F")

def main():
      some_ifs(1, 2, 3)
main()
```

(2 marks)

ID: .....

c) Complete the main() function of the following program. The program first gets the number of hours worked from the user, then gets the age from the user, then the salary is calculated and finally the program displays both the salary and the hours. All the functions for this program have already been defined. You are required to complete the main() function of the program by writing FOUR lines of code, with each line making a call to one of the three defined functions. Below are three separate example outputs using the completed program (the user input is shown in a larger font and in bold).

```
Enter hours worked: 6
Enter age: 34
Salary: $153.0  (6 hours)
```

```
Enter hours worked: 10
Enter age: 15
Salary: $191.25  (10 hours)
```

```
Enter hours worked: 3
Enter age: 20
Salary: $76.5  (3 hours)
```

```
def get_salary(num_hours, age):
    pay_per_hour = 25.5
    youth_rate = 0.75

    pay = num_hours * pay_per_hour
    if age < 16:
        return pay * youth_rate

    return pay

def display_results(num_hours, salary):
    hours_worked = "  (" + str(num_hours) + " hours)"
    print()
    print("Salary: $", salary, hours_worked, sep = "")

def get_user_input(prompt):
    return int(input(prompt))

def main():
```

(6 marks)

```
main()
```

ID: .....

Question 3 (10 marks)

a) Give the output produced by the following code.

```
num1 = 5
num2 = 9
count = 0
while num1 < num2:
      if num2 - num1 >= 3:
          count += 1
      print(num1, num2, count)
      num1 = num1 + 2
      num2 = num2 + 1
```

(2 marks)

b) Give the output produced by the following code.

```
original = 40
result = 0
for num in range(4, 16, 4):
      print(result, end = " ")
      result = original - num
      original = original + 2

print()
print(original)
```

(2 marks)

ID: .....

c) Complete the `get_terms_total(num_terms)` function which is passed one parameter, the number of terms to be totalled. The function returns the total of the following series of fractions:

$$\frac{1}{3} + \frac{2}{5} + \frac{3}{7} + \frac{4}{9} + \frac{5}{11} + \frac{6}{13} + \ldots$$

The first number in the series is 1/3, the next number is 2/5, the next number is 3/7, and so on, i.e., for each term, the top number of the fraction increases by 1 and the bottom number increases by 2. The total returned by the function is rounded to two decimal places. For example, the following program using the completed `get_terms_total()` function prints:

```
Total of 2 terms: 0.73
Total of 4 terms: 1.61
Total of 7 terms: 2.99
```

```
def get_terms_total(num_terms):
```

(6 marks)

```
def main():
    print("Total of 2 terms:", get_terms_total(2))
    print("Total of 4 terms:", get_terms_total(4))
    print("Total of 7 terms:", get_terms_total(7))

main()
```

ID: .....

Question 4 (10 marks)

a) Given the following code:

```
object1 = [3, 4.5, True, "False", 4 * 3 + 1]
object2 = object1[3]
object3 = object1[2]
```

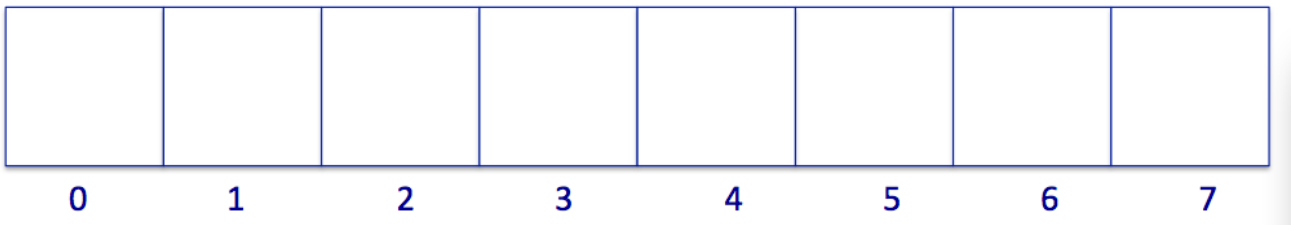
what is the type of the three Python objects: `object1`, `object2` and `object3`?

```
object1 is of type:

object2 is of type:

object3 is of type:
```

(1.5 marks)

b) In the boxes below, show each element of `a_list` after the following code has been executed. Use as many of the boxes as you need.

```
a_list = [5, 3, 8, 9, 1]

a_list[0] = a_list[1] + a_list[3]
a_list[2] = a_list[3] * a_list[1]
a_list = a_list + [a_list[3] - a_list[4]]
```



(2.5 marks)

ID: .....

c) In the following program, complete the `get_not_in_both_lists()` function which has two list parameters. The function returns a new list which contains all the elements of `list1` (the first parameter) which are not in `list2` (the second parameter), followed by all the elements in `list2` which are not in `list1`. For example, the completed program gives the output:

```
List1: [5, 3, 8, 9, 1]
List2: [5, 9, 3, 2]
List3: [8, 1, 2]
```

```
def get_not_in_both_lists(list1, list2):
```

(6 marks)

```
def main():
    list1 = [5, 3, 8, 9, 1]
    list2 = [5, 9, 3, 2]
    list3 = get_not_in_both_lists(list1, list2)
    print("List1:", list1)
    print("List2:", list2)
    print("List3:", list3)

main()
```

ID: .....

Question 5 (10 marks)

a) Using the code tracing technique shown in lectures, perform a code trace for the following program and give the output. Give the output in the space below and **show the code trace in the space provided on the next page**.

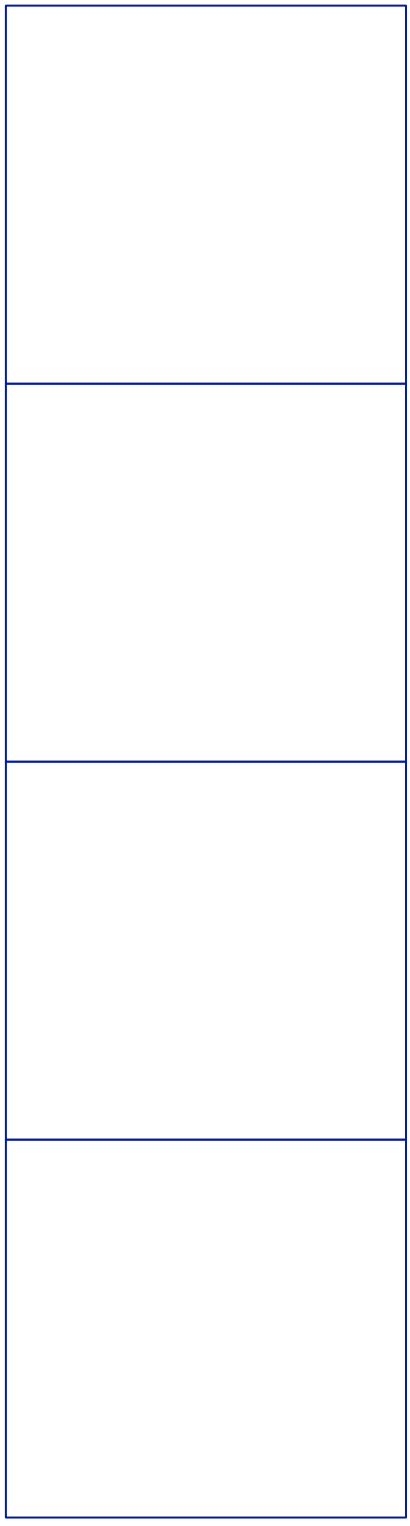
```
def main():
      a = 1
      b = 2
      b = first(a, b)
      print("4.", b)

def first(n1, n2):
      a = second(n1, n2, 3)
      print("2.", a)
      b = second(n2, n1, 2)
      print("3.", b)
      return a + b

def second(num1, num2, num3):
      print("1.", num1, num2)
      value = str(num1) + str(num2)
      value = int(value)
      return value * num3

main()
```

Give the output:

(4 marks)

ID: .....

Show the code trace in the space below:



(6 marks)