|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Part** | **1** | **2** | **3** | **4** | **Total** |
| *maximum* | **25** points | **25** points | **25** points | **25** points | **100**G101010 pointsG |
| ***Your Score*** |  |  |  |  |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Sets and Dictionaries**

Reading Assignment: Thoroughly read Chapter 11 in the course textbook.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 1 Glossary Terms**

Define, in detail, each of these glossary terms from the realm of computer programming logic and design and computer topics, in general. If applicable, use examples to support your definitions. Consult your notes or course textbook(s) as references or by visiting Web sites such as: [**http://www.ask.com**](http://www.ask.com),[**http://www.bing.com**](http://www.bing.com), [**http://www.webopedia.com**](http://www.webopedia.com)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**(a) Array of Buckets**

|  |
| --- |
| An array of buckets (linked lists) is used in the collision processing strategy known as “chaining” to ensure that hashed indices of items in a set or dictionary do not overwrite or collide with one another, AND ensures that space is used efficiently.  It also allows for O(1) access, insertion, and removal of items whose indices may not be integers. This makes it so we can e.g., use dictionaries with random O(1) access, since the lookup is on the hashed index. |

**(b) Hashing**

|  |
| --- |
| Hashing is creating a sort of code based on an input. It can involve various levels of complexity to ensure randomness, however, each input must have the same hash value to fit into a hash table.  When hashing for a set or dictionary, the index of an item’s position is that item’s unique hash. This enables us to access items whose keys are strings, for example, at O(1) constant time. |

**(c) Key - Value Pairs**

|  |
| --- |
| A key value pair is a key (e.g., a hash) and a data item, and are typically used in the context of dictionaries. |

**(d) Linear Collisions**

|  |
| --- |
| A linear collision takes place with an items hashed position index is the same as an existing items. There are various methods to deal with this, the best of which is “Chaining”. |

**(e) Quadratic Probing**

|  |
| --- |
| Quadratic probing is another method for hashing, and involves incrementing an index quadratically (e.g., i^2) until it finds an empty cell. This can leave a lot of empty space, however… Chaining is best. |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 2 True / False Exercises**

For each of these exercises, enter True or False in the spaces provided.

**FALSE** **(1)** Much like a list, a set contains items that are in a particular order.

**TRUE** **(2)** With a set, the difference and subset operations are not symmetric.

**FALSE** **(3)** If S1 and s2 are sets, the expression s1.issubset(s2) returns True if s2 is a subset of s1.

**TRUE (4)** A set is similar to a bag, but it contains unique data items and additional methods.

**TRUE (5)** A hashing function acts on a given key by returning its absolute position in an array.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 3 Multiple Choice Exercises**

Select the correct response or responses.

**(1)** Which of the following is true about sets?

(a) the items in a set are arranged in order

(b) the difference and subset operations on a set are symmetric

**(c) there are no duplicate items in a set**

(d) there is no standard set class in Python

**(2)** Which of the following is a subset of Set A if Set A is {19 4 26 8} ?

**(a) {}**

(b) {19 4 26 8 0}

(c) {4 8 19 26 44}

(d) None of the choices are subsets of Set A

**(3)** What is the value of set S after the following operations?

**S = set([3, 9, 6])**

**S.add(6)**

**S.add(4)**

**S.remove(6)**

(a) {3 9 6 4} **(b) {3 9 4}**

(c) {3 9 6} (d) {3 4 6 9}

**(4)** What strategy for implementing sets attempts to approximate random access into an array for insertions, removals and searches?

(a) indexing (b) linking

**(c) hashing** (d) keying

**(5)** What is the performance value of the array - based implementations of sets and dictionaries?

(a) *O* ( *n* 2 ) **(b) *O* ( *n* )**

(c) *O* *n* (d) *O* ( 1 )

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 4 Programming Exercises**

Python Dictionary Methods

**clear() copy()**

**fromkeys() get()**

**items() keys()**

**values() pop()**

**popitem() setdefault()**

**update()**

**(1)** The methods available to dictionaries are shown above.

Consider the program that follows, which incorporates dictionaries and the methods that are accessible to them. Review the following code statements and then perform each of these duties:

(a) Make certain that all of the above dictionary methods are included in the given program. If not, supplement the program with the appreciate use of any missing methods. You can choose where the statement(s) for the method are placed.

(b) Enhance the program with comment statements that illustrate how or why each dictionary method is utilized.

(c) For the last action of the program, at the bottom of the code statements,  
 sum all of the values in the dictionary named as **dictionary**.

|  |
| --- |
| dictionary = {"a": 4, "b": 5, "c": [6, 7]}  dictionary\_1 = {"a": 8, "m": 2, "v": 7}  *# update dictionary method updates a dictionary with all the key/values in another dictionry, however it does not overwrite any existing key/value pairs*  dictionary.update(dictionary\_1)  *# returns a key/value pair as a tuple. Pairs are returned in LIFO order.*  x = dictionary.popitem()  *# pops out a key/value pair based on key lookup*  x = dictionary.pop("b")  *# dict.keys() returns an iterator over the dictionary keys*  *# for key in dictionary.keys():*  *# print(key)*  *# dict.values() returns an iterator over the dictionary values*  *# for value in dictionary.values():*  *# print(value)*  *# dict.items() returns an iterator over the dictionary key/value pairs*  *# for key, value in dictionary.items():*  *# print(key, "-", value)*  *# get returns the value at the key, here, it returns NONE because we popped out 'b'*  x = dictionary.get("b")  *# fromkeys creates a new dictionary from the keys in another dictionary or other iterable, and fills in a value*  dictionary\_1 = dict.fromkeys(dictionary, 1)  *# copy returns a shallow copy of a dictionary*  dictionary\_1 = dictionary.copy()  *# assignments uses the \_\_set\_\_ method, assigns a value to a key input*  dictionary\_1["b"] = 2  *# clear clears the dictionary, returns empty {}*  dictionary.clear()  *# length returns the number of entries in a dict*  length = len(dictionary)  *# MISSING setdefault, setdefault returns a key if it's in the dictionary,*  *# else will set the key/value pair with the given input*  dictionary.setdefault("testDefaultKey", "test")  *# summing all values of dictionary\_1*  sum = 0  *for* value *in* dictionary\_1.values():  *if* type(value) == list:  *for* i *in* value:  sum += 1  *else*:  sum += value  print(sum)  *# returns 14* |