Exam 3 Solutions

Fundamentals

Write ONLY what is requested.

a. What is a hash value?
Solution:
A hash value is the return value of a hash function

b. What is a map data structure?
Solution:
A map is a collection of key-value pairs such that all keys are distinct

c. What is the meaning of a collision in hashing?
Solution:
A collision in hashing means two or keys have the same hash value

d. Which sorting algorithm performs the least amount of swaps for the worst-case scenario, and how many swaps will it perform if the array has a size of n?
Solution:
The selection sort performs the least amount of swaps for the worst-case scenario and it will perform n swaps.

e. How does the open-addressing insertion method deal with collisions?
Solution:
the insertion method for open-addressing moves a key to the next available slot whenever a collision occurs.

Runtime

Construct the runtime table that includes a statement column and determine the runtime functions of the following function for the worst-case scenario. Let the cost of every operation be 1. Write the function in terms of n, which is the size of the array. You may need to use the ceiling or floor function for an accurate solution.

```
void D(Array<string>& data)
{
  for(int i = 0;i < data.Size();i += 1)
  {
    data[i] = "[";

    for(char j = '0';j != '9';j += 1)
    {
      data[i] += j;
    }
    data[i] = "]";
  }
}
```

Solution:

statement	cost	time
`int i = 0`	1	1
`i < data.Size()`	1	$n+1$
`data[i] = "[";`	1	n
`char j = '0'`	1	$\sum_{i=1}^{n} 1$
`j != '9'`	1	$\sum_{i=1}^{n} 10$
`data[i] += j;`	1	$\sum_{i=1}^{n} 9$
`j += 1`	1	$\sum_{i=1}^{n} 9$
`data[i] = "]";`	1	n
`i += 1`	1	n

$$\mathbf{T(n)} = 4n + 2 + \sum_{i=1}^{n} 29 = 33n + 2$$

Tracing

Write an array trace table for any two of the three sorting algorithms discussed in class [bubble sort, insertion sort, selection sort] that provides only the swaps that will be performed on the array data = $[\mathbf{2}, \mathbf{3}, \mathbf{7}, \mathbf{4}, \mathbf{8}, \mathbf{1}]$. Each trace table must start with the initial value of data.

Solution:

Bubble Sort	Insertion Sort	Selection Sort
[2,3,7,4,8,1]	[2,3,7,4,8,1]	[2,3,7,4,8,1]
[2,3,4,7,8,1]	[2,3,4,7,8,1]	[1,3,7,4,8,2]
[2,3,4,7,1,8]	[2,3,4,7,1,8]	[1,2,7,4,8,3]
[2,3,4,1,7,8]	[2,3,4,1,7,8]	[1,2,3,4,8,7]
[2,3,1,4,7,8]	[2,3,1,4,7,8]	[1,2,3,4,7,8]
[2,1,3,4,7,8]	[2,1,3,4,7,8]	
[1,2,3,4,7,8]	[1,2,3,4,7,8]	

Problem Solving

Write the void function `InsertionSort()` whose header is

```
template <typename T>
void InsertionSort(Node<T>* root)
```

Its definition is the insertion sort algorithm implemented with a linked list. Remember a linked list can be empty.

Solution:

```
template <typename T>
void InsertionSort(Node<T>* root)
{
  if(root != NULL)
  {
    for(Node<T>* i = root->next;i != NULL;i = i->next)
    {
      Node<T>* j = i;

      while(j != NULL && j->data > j->prev->data)
      {
        Swap(j->data,j->prev->data);
        j = j->prev;
      }
    }
  }
}
```

Implementation

Given that the fields of a class named *HashMap* is

```
template <typename V>
class HashMap
{
  Node<Pair<int,V>>* slots[200];
};
```

write the following methods

- private int method `hash()` that takes an int parameter. It should implement the division method algorithm for hash functions using the absolute value of the parameter.
  Solution:

```
int hash(int key)
{
  return ((key < 0)?(-1 * key):(key))% 200;
}
```

- **public bool method named** `Contains()` **that takes an int parameter named** ***key*****. It returns true if** ***key*** **is in the hashmap; otherwise, it returns false.**

  Solution:

```
void Contains(int key)
{
  Node<Pair<int,V>>* t = slots[hash(key)];

  while(t != NULL && t->data->key != key)
  {
    t = t->next;
  }
  return (t != NULL);
}
```