CS 32 Worksheet Week 6

Concepts: Recursion, Templates, STL

Recursion Problems

1. (10 min) Implement the function `getMax` recursively. The function returns the maximum value in `a`, an integer array of size `n`. You may assume that `n` will be at least

2. (15 min) Rewrite the following function recursively. You can add new parameters and completely change the function implementation, but you can't use loops.

This function sums the elements of an array of nonnegative numbers from left to right until the sum exceeds some threshold. At that point, the function returns the running sum. Returns -1 if the threshold is not exceeded before the end of the array is reached.

```
int sumOverThreshold(int x[], int length, int threshold)
  int sum = 0;
  for(int i = 0; i < length; i++) {
    sum += x[i];
      if (sum > threshold) {
        return sum;
      }
  }
  return -1;
}
```

3. (15 min) Given a string *str*, recursively compute a new string such that all the ‘x’ chars have been moved to the end.
Hint: [*https://www.cplusplus.com/reference/string/string/substr/*](https://www.cplusplus.com/reference/string/string/substr/)

```
string endX(string str);
```

Example:

```
endX("xrxe") → "rexx"
```

4. (15 min) Implement the following function in a recursive fashion:

```
bool isSolvable(int x, int y, int c);
```

This function should return true if there exists nonnegative integers *a* and *b* such that the equation *ax* + *by* = *c* holds true. It should return false otherwise.

Examples:

```
isSolvable(7, 5, 45) == true     //a == 5 and b == 2
isSolvable(1, 3, 40) == true     //a == 40 and b == 0
isSolvable(9, 23, 112) == false
```

Template/STL Problems

5. (5 min) The following code has 3 errors that cause either runtime or compile time errors. Find and fix all of the errors.

```
class Potato {
public:
  Potato(int in_size) : size(in_size) { };
  int getSize() const {
    return size;
  };
private:
  int size;
};

int main() {
  vector<Potato> potatoes;
  Potato p1(3);
  potatoes.push_back(p1);
  potatoes.push_back(Potato(4));
  potatoes.push_back(Potato(5));

  vector<int Potato>::iterator it = potatoes.begin();
  while (it != potatoes.end()) {
    potatoes.erase(it);
    it++;
  }

  for (it = potatoes.begin(); it != potatoes.end(); it++) {
    cout << it.getSize() << endl;
  }
}
```

6. (15 min) Implement a stack class *Stack* that can be used with any data type using templates. Use a linked list (not an STL `list`) to store the stack and implement the functions *push(), pop(), top(), isEmpty(),* a default constructor, and a destructor that deletes the linked list nodes.

7. (15 min) Implement a vector class *Vector* that can be used with any data type using templates. Use a dynamically allocated array to store the data. Implement only the *push_back()* function, default constructor, and destructor.

Extra Practice

Recursion:

1. (10 min) Implement the function `sumOfDigits` recursively. The function returns the sum of all of the digits in the given *positive* integer `num`.
Hint: Use integer division

```
int sumOfDigits(int num);

sumOfDigits(176); // return 14
sumOfDigits(111111); // return 6
```

2. (15 min) Write the following linked list functions recursively.

```cpp
// Node definition for singly linked list
struct Node {
  int data;
  Node* next;
};

// inserts a value in a sorted linked list of integers
// returns list head
// before: 1 → 3 → 5 → 7 → 15
// insertInOrder(head, 8);
// after: 1 → 3 → 5 → 7 → 8 → 15
Node* insertInOrder(Node* head, int value);

// deletes all nodes whose keys/data == value, returns list head
// use the delete keyword
Node* deleteAll(Node* head, int value);

// prints the values of a linked list backwards
// e.g. 0 → 2 → 1 → 4 → 1 → 7
// reversePrint(head) will output 714120
void reversePrint(Node* head);
```

Template/STL:

1. (5 min) Will this code compile? If so, what is the output? If not, what is preventing it from compiling?

Note: We did not use `namespace std` *because* `std` *has its own implementation of* `max` *and* `namespace std` *will thus confuse the compiler.*

```
template <typename T>
T max(T x, T y) {
  return (x > y) ? x : y;
}

int main() {
  std::cout << max(3, 7) << std::endl;     // line 1
  std::cout << max(3.0, 7.0) << std::endl; // line 2
  std::cout << max(3, 7.0) << std::endl;   // line 3
}
```

2. (20 min) You are given an STL `set<list<int>*>`. In other words, you have a set of pointers, and each pointer points to a list of ints. Consider the sum of a list to be the result of adding up all elements in the list. If a list is empty, treat its sum as zero. Write a function that removes the lists with odd sums from the set. The lists with odd sums should be deleted from memory and their pointers should be removed from the set. This function should return the number of lists that are removed. You may assume that none of the pointers is null.