String to Integer Write Up

For this problem, I wrote a solution to convert a string into an integer. This solution needed to account for a number of test cases, namely, leading zeros, a negative/plus sign, whitespace, miscellaneous characters, and integers larger than the 32-bit signed integer range. Throughout the problem 4 different boolean variables are used to keep track of where we are at in the string, so that the program can account for the various test cases. The boolean variables used test if a 0 character is encountered is a leading zero (leadingZero), if the number is negative (neg), if this is the first non-whitespace character encountered (first), and if a digit has been encountered in the string yet (digit). To solve this problem, I started off with a large while loop that iterates through each character in the string. I first check for 3 cases for the current character, is it a space, digit, or neither? If it is a space and there has not yet been a digit in the string, I will delete the space from the string. Otherwise, I will delete from the space onward and break out of the while loop. If the character is a digit, I will check if it is a leading 0. If it is a leading 0, I will delete it from the string. Otherwise, I will keep the digit in the string for later conversion and increment my index counter by 1. If the character is not a digit but we have already encountered a digit character, then we will delete from that character onward and exit the loop. If the first non-whitespace character is not a digit or +/- or if we are not on our first +/- and the character is not a digit or space, then the string is considered to have no valid conversion and we will replace the string with 0 and break out of the loop. When we encounter a '-' as our first non-whitespace character, we will set a boolean variable "neg" to true to symbolize that the number in the string is negative. Every non-digit character will be erased for simplicity's sake. After we have our final digit ready in our string, we will check if our index counter (the length of the string) reached larger than 10. If so, then we will automatically know that the number in the string was too large and we will assign our final number to INT-MIN if it was negative, INT-MAX otherwise. If the length is exactly 10, we will check in more depth if the number is larger than INT-MIN or INT-MAX. Otherwise, we will simply convert the string to an integer using the atoi function and return our final number. The runtime complexity of this problem is O(n), with n being the size of the string. This is due to our while loop which will iterate through each character in the string, performing a few lower order operations (O(1)) with each iteration. The space complexity of this solution is O(1), as we only allocate memory for a few variables that take up a constant amount of space. Our memory does not grow according to the size of our string as we simply make changes to the same string before converting it to an integer.

```
class Solution {
public:
   int myAtoi(string str)
   {
      int num=0;
      int i=0;
      bool neg=false;
      bool first=true;
      bool digit=false;
      bool leadingZero=true;

      while(i<str.size())
      {
         if(isspace(str[i]))
         {
            if(first==false)
```

```
    {
        str.erase(i);
        break;
    }
    else
    {
        str.erase(i, 1);
    }
}
else if(isdigit(str[i]))
{
    first=false;
    digit=true;

    if(str[i]=='0' && leadingZero==true)
    {
        str.erase(i,1);
    }
    else
    {
        leadingZero=false;
        i++;
    }
}
else
{
    if(digit==true)
    {
        str.erase(i);
        break;
    }
    if(first==true && str[i]!='+' && str[i]!='-')
    {
        str.erase();
        str="0";
        break;
    }
    else if(first==false && digit==false)
    {
        str.erase();
        str="0";
        break;
    }
    else if(str[i]=='-' && first==true)
    {
        neg=true;
        first=false;
    }
    else if(str[i]=='+' && first==true)
```

```
        {
            first=false;
        }

        str.erase(i, 1);
    }

}
if(i>10)
{
    if(neg)
        {
            num=INT_MIN;
        }
        else
        {
            num=INT_MAX;
        }
}
else if(i==10)
{
    if((int)(str[0]-'0')>2)
    {
        if(neg)
        {
            num=INT_MIN;
        }
        else
        {
            num=INT_MAX;
        }
    }
    else if((int)(str[0]-'0')==2)
    {
        int num2=atoi((str.substr(1)).c_str());

        if(num2>=147483647 && neg==false)
        {
            num=INT_MAX;
        }
        else if(num2>=147483648 && neg==true)
        {
            num=INT_MIN;
        }
        else
        {
            num=atoi(str.c_str());

            if(neg)
```

```
                {
                    num=num*-1;
                }
            }

        }
        else
        {
            num=atoi(str.c_str());

            if(neg)
            {
                num=num*-1;
            }
        }
    }
    else
    {
        num=atoi(str.c_str());

        if(neg)
        {
            num=num*-1;
        }
    }

    return num;
  }
};
```