CS1010 Tutorial 6

Ryan Tan Yu

Wednesday, 10th March 2021

Agenda for Today

- Problem Set 16
- Assignment 3 (Comments)
- Assignment 4 (Tips)

Problem Set 16

Problem Set 16

- Write the following functions without calling the standard C library functions declared in `string.h`
  1. `long string_length(char str[])`
     Returns the number of *characters* of the string `str`
  2. `bool string_equal(char str1[], char str2[])`
     Returns true if `str1` and `str2` contain exactly the same content
  3. `char *string_in_string(char needle[], char haystack[])`
     Returns a pointer to the first character of the first occurrence of `needle` in `haystack` if found. If `needle` does not occur anywhere in `haystack`, return `NULL`. If `needle` is an empty string, `haystack` is returned.

Problem Set 16.1: String Length

Main idea: Loop through from start of string until \0 is reached. Keep an incrementing counter as we loop. Note that we *don't* count the nul-terminating character

```
long string_length(char str[])
{
    long count = 0;
    long index = 0;
    while (str[index] != '\0') {
        count += 1;
    }
    return count;
}
```

Problem Set 16.2: String Equals

Check if two strings are equal. The idea is to loop from the start to the end of both strings and check the characters as we loop.

Some gotchas to consider:

- Consider the case if the two strings are empty
- Consider the case if one string is a substring of the other

Problem Set 16.2: String Equals

```
bool string_equal(char str1[], char str2[]) {
    long i = 0;
    long j = 0;
    while (str1[i] != '\0' && str2[j] != '\0') {
        if (str1[i] != str2[j]) {
            return false;
        }
        i += 1;
        j += 1;
    }
    if (str1[i] == '\0' && str2[j] == '\0') {
        return true;
    }
    return false;
}
```

Problem Set 16.3: String in String

Known as `strstr` in the standard library of C. Let s be the containing string, and t be the substring to be checked for

We return

- a pointer to the first character of the first occurrence of t in s if t is found in s
- `NULL` if t cannot be found in s
- t if $s = \epsilon$

Problem Set 16.3: String in String

s is the containing string, t is the substring, and $|s| = n$ and $|t| = k$

We need 2 functions

- To check if each $s[i...i + k] = t$ for some i
- To loop through each $s[i]$ and call the above function

(Note: this algorithm is slow and inefficient and runs in $O(nk)$. It turns out that we can do this in $O(n)$ time using the KMP Algorithm)

Problem Set 16.3: String in String

```
bool has_needle_here(char needle[], char haystack[]) {
    long index1 = 0;
    long index2 = 0;
    while (needle[index1] != '\0') {
        if (needle [index1] != haystack[index2]) {
            return false;
        }
        index1 += 1;
        index2 += 1;
    }
    return true;
}
```

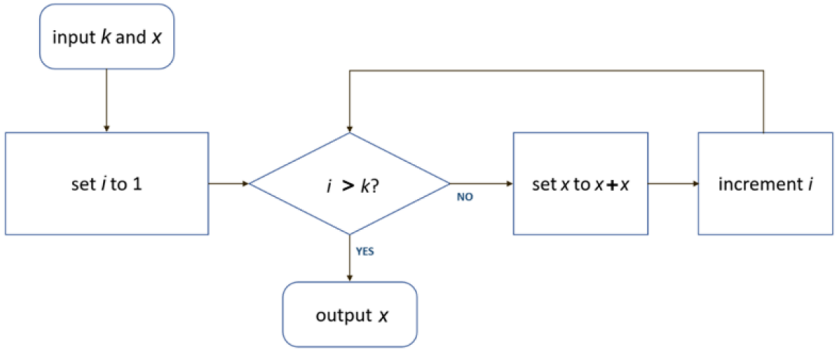
Problem Set 16.3: String in String

```
char *string_in_string(char needle[], char haystack[]) {
    if (haystack[0] == '\0') {
        return needle;
    }

    long start = 0;
    char end = string_length(haystack) - string_length(needle);

    while (start < end) {
        if (has_needle_here(needle, &haystack[start]) {
            return &haystack[start];
        }
        start += 1;
    }
```

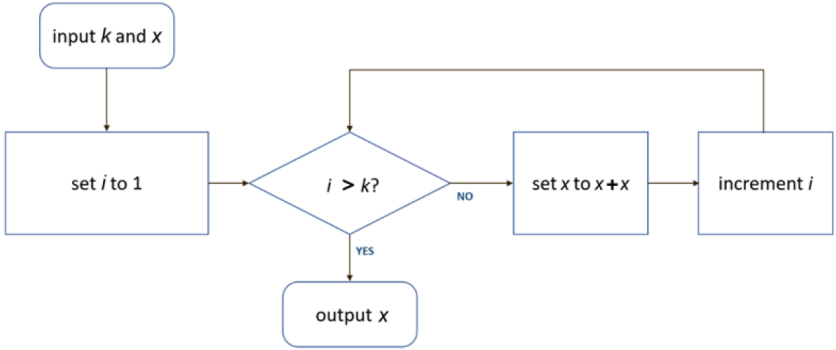
Midterm Discussion

Question 2 - Flowcharts

Determine what the algorithm calculates (assuming that both k and x are integers).



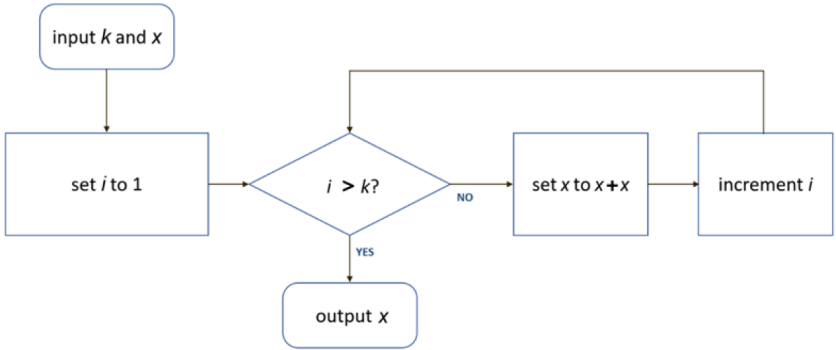
Question 2 - Flowcharts



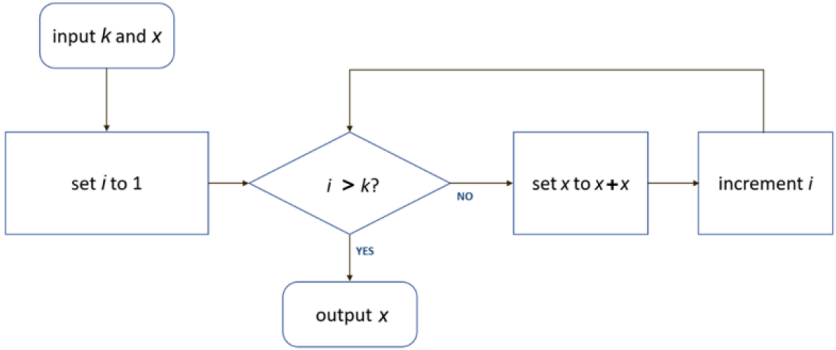
The algorithm calculates the value $x \cdot 2^k$

Question 3 - Valid Input Ranges

Specify the value ranges for k and x such that the output is correctly calculated based on your answer above in Question 2.

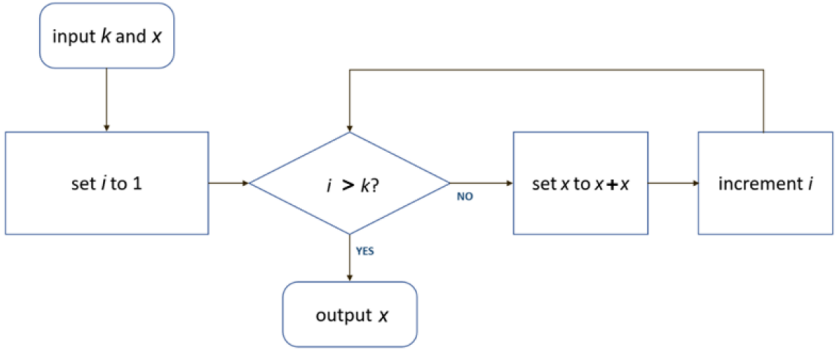


Question 3 - Valid Input Ranges



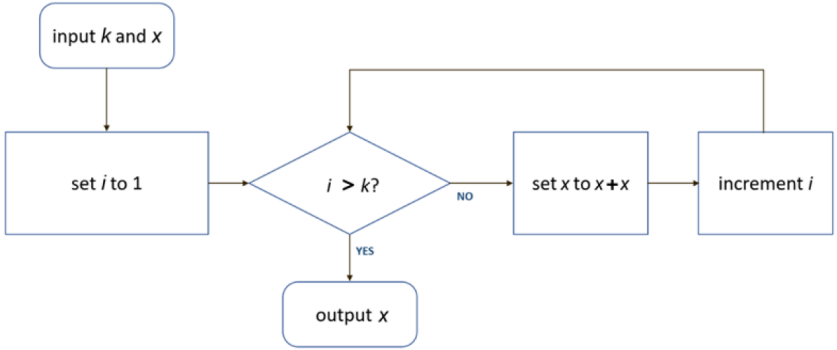
- The algorithm definitely works for positive k
- What about $k = 0$?
- What about negative k?

Question 3 - Valid Input Ranges



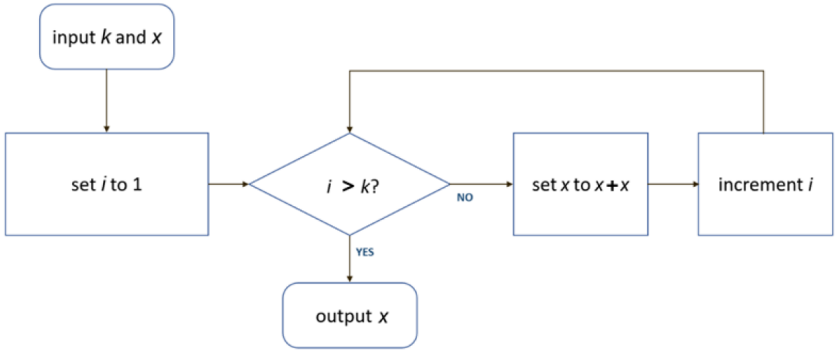
- For $k = 0$, the algorithm outputs $x \cdot 2^0 = x$
- Say $k = -1$, then the algorithm should output $x \cdot 2^{-1} = \frac{1}{2}x$
  - The algorithm outputs x instead, we can reasonably deduce that $k \geq 0$ is the valid range of k

Question 3 - Valid Input Ranges



- The algorithm definitely works for positive x
- For $x = 0$ and $k \geq 0$, the algorithm is correct.
- What about $x < 0$?

Question 3 - Valid Input Ranges



- For $x < 0$, $x \cdot 2^k$ must output a negative number for $k \geq 0$
- x at any iteration of the loop will always be negative
- Try $x = -1$ and $k = 2$, then $-1 \cdot 2^2 = -4 \rightarrow$ the algorithm outputs -4
- Try $x = 3$ and $k = 3$, then $3 \cdot 2^3 = 24 \rightarrow$ the algorithm outputs -24
- We can reasonably deduce that the valid range for x is any integer

Question 4 - Boolean Expression

Consider the following two logic expressions.

- `(year % 4 == 0) && (year % 100 != 0)`
- `!((year % 4 == 0) && (year % 100 != 0))`

Are the two logical expressions equivalent? Justify your answer.

Question 4 - Boolean Expression

Consider the following two logic expressions.

- `(year % 4 == 0) && (year % 100 != 0)`
- `!((year % 4 != 0) && (year % 100 == 0))`

We apply De Morgan's Law on the second boolean expression.

```
    !((year % 4 != 0) && (year % 100 == 0))
==> !(year % 4 != 0) || !(year % 100 == 0)
==> year % 4 == 0 || year % 100 != 0
```

This is the same as the first statement, therefore they are equivalent.

Question 5 - Tracing an Algorithm

"The code below returns the sum of integers n $[1, n]$". True or false? Justify your answer.

```
long sum(long n)
{
    long result = 0;
    for (long i = 1; i = n; i += 1) {
        result += i;
    }
    return result;
}
```

Question 5 - Tracing an Algorithm

"The code below returns the sum of integers n $[1, n]$". True or false? Justify your answer.

```
long sum(long n)
{
    long result = 0;
    for (long i = 1; i = n; i += 1) {
        result += i;
    }
    return result;
}
```

- Notice that the condition $i = n$
- Assignment expressions return the right-hand-side of the expression
- Therefore the loop will terminate indefinitely for positive n
- The condition should be `i <= n`
- The statement is false.

Question 6 - Tracing an Algorithm II

Consider the following C program

```
1   long f(long k)
2       return k * 10;
3
4   long g(long x)
5       return x % 100;
6
7   int main()
8       long k = 123;
9       long x = f(k);
10      long y = g(k);
11      print(f(g(x + y)));
```

When executed, the above program will print _____.

Question 6 - Tracing an Algorithm II

```
1   long f(long k)
2       return k * 10;
3
4   long g(long x)
5       return x % 100;
6
7   int main()
8       long k = 123;
9       long x = f(k);
10      long y = g(k);
11      print(f(g(x + y)));
```

- At line 9, $x = f(123) = 123 \cdot 10 = 1230$
- At line 10, $y = g(123) = 123 \bmod 100 = 23$
- At line 11, we work inside outside
  - $x + y = 1230 + 23 = 1253$
  - $g(x + y) = g(1253) = 53$
  - $f(g(x + y)) = f(53) = 530$

Question 7 - Tracing an Algorithm III

Consider the following C program

```
long f(long n) {
    if (n < 2) {
        return 1;
    } else if (n % 2 == 0) {
        return 1 + f(n / 2);
    } else {
        return 2 + f(n - 1);
    }
}
```

When executed with $f(10)$, the above program will print _____.

Question 7 - Tracing an Algorithm III

```
long f(long n) {
    if (n < 2) {
        return 1;
    } else if (n % 2 == 0) {
        return 1 + f(n / 2);
    } else {
        return 2 + f(n - 1);
    }
}
```

$$
\begin{aligned}
f(10) &= 1 + f(5) \\
&= 1 + 2 + f(4) \\
&= 1 + 2 + 1 + f(2) \\
&= 1 + 2 + 1 + 1 + f(1) \\
&= 1 + 2 + 1 + 1 + 1 \\
&= 6
\end{aligned}
$$

Question 8 - Tracing an Algorithm IV

Consider the following code snippet

```
long a = 10;
long b;

while (a > 0) {
    b = a / 2;
    if (a * b > a + b) {
        cs1010_println_long(b);
    }
    a -= 3;
}
```

When executed, the above program will print ____.

Question 8 - Tracing an Algorithm IV

```
long a = 10;
long b;

while (a > 0) {
    b = a / 2;
    if (a * b > a + b) {
        cs1010_println_long(b);
    }
    a -= 3;
}
```

- First iteration
  - $a = 10$
  - $b = 5$
  - $a \cdot b = 50$ and $a + b = 15$
  - **print** $b = 5$
  - $a = 7$
- Second iteration
  - $a = 7$
  - $b = 3$
  - $a \cdot b = 21$ and $a + b = 10$
  - **print** $b = 3$
  - $a = 4$

Question 8 - Tracing an Algorithm IV

```
long a = 10;
long b;

while (a > 0) {
    b = a / 2;
    if (a * b > a + b) {
        cs1010_println_long(b);
    }
    a -= 3;
}
```

- Third iteration
  - $a = 4$
  - $b = 2$
  - $a \cdot b = 8$ and $a + b = 6$
  - **print** $b = 2$
  - $a = 1$
- Fourth iteration
  - $a = 1$
  - $b = 0$
  - $a \cdot b = 0$ and $a + b = 1$
  - $a = -2$
- *Exit Loop*

Question 9 - Simplification of Boolean Conditionals

Consider the following nested `if` statements for variables `day` in $[1, 7]$ and `time` from $[0, 23]$

```
if (day <= 5) {
    if (time <= 8) {
        cs1010_println_string("Case␣1");
    } else {
        cs1010_println_string("Case␣2");
    }
} else if (day == 6) {
    if (time <= 8) {
        cs1010_println_string("Case␣1");
    } else {
        cs1010_println_string("Case␣4");
    }
} else {
    if (time <= 8) {
        cs1010_println_string("Case␣3");
    } else {
        cs1010_println_string("Case␣4");
    }
}
```

Question 9 - Simplification of Boolean Conditionals

Write the logical expressions `EXP_A`, `EXP_B`, `EXP_C` for the following `if` statement (without changing the rest of the code) so that it is equivalent to the given `if` statement.

```
if (EXP_A) {
    print Case 1;
} else if (EXP_B) {
    print Case 2;
} else if (EXP_C) {
    print Case 3;
} else {
    print Case 4;
}
```

Question 9 - Simplification of Boolean Conditionals

The strategy is to create a *table* of value, similar to **Taxi** or **Twilight** or **Parking Fee**

	time <= 8	time > 8
day <= 5	Case 1	Case 2
day == 6	Case 1	Case 4
day == 7	Case 3	Case 4

Now we can infer the cases and their corresponding `day` and `time` values

- `EXP_A: (day <= 6) && (time <= 8)`
- `EXP_B: (day <= 5) && (time > 8)`
- `EXP_C: (day == 7) && (time <= 8)`

Question 10 - Assertions

Consider the following function such that $x < y$,

```
1  long f(long x) {
2      long res = 0;
3      long a = x;
4      while (a > 0) {
5          a -= 2;
6          res += 1;
7      }
8      return res;
9  }
```

Give an assertion related to (only) *a* between Line 5 and 6

Question 10 - Assertions

Give an assertion related to (only) *a* between Line 5 and 6

```
1  long f(long x) {
2      long res = 0;
3      long a = x;
4      while (a > 0) {
                {a > 0}
5          a -= 2;
             {a > -2} // the assertion
6          res += 1;
7      }
8      return res;
9  }
```

Question 11 - Loop Invariants

Specify the loop invariant that relates `res`, a and x

```
1  long f(long x) {
2      long res = 0;
3      long a = x;
4      while (a > 0) {
5          a -= 2;
6          res += 1;
7      }
8      return res;
9  }
```

Question 11 - Loop Invariants

Specify the loop invariant that relates `res`, a and x

```
1  long f(long x) {
2      long res = 0;
3      long a = x;
4      while (a > 0) {
5          a -= 2;
6          res += 1;
7      }
8      return res;
9  }
```

1. Figure out what you want to be true at the end of the loop
2. Try to reach something similar with your invariant

Question 11 - Loop Invariants

Specify the loop invariant that relates `res`, a and x

```
1  long f(long x) {
2      long res = 0;
3      long a = x;
         {res = (x - a) / 2}
4      while (a > 0) {
5          a -= 2;
6          res += 1;
         {res = (x - a) / 2}
7      }
         {res = (x - a) / 2}
8      return res;
9  }
```

Question 12 - Writing Nested Loops

Write a code snippet using for loops and the CS1010 I/O library functions to print the pattern below:

```
1*2*3*4*5*6*7*8*9*
3*4*5*6*7*8*9*
5*6*7*8*9*
7*8*9*
9*
```

Question 12 - Writing Nested Loops

```
1*2*3*4*5*6*7*8*9*
3*4*5*6*7*8*9*
5*6*7*8*9*
7*8*9*
9*
```

- This question is about recognising patterns, like Assignment 2's "Patterns"
- It's pretty simple to see - the starting number of each row follows $1, 3, 5, 7, \ldots$
- After the starting number, print out `i*` from $[i, 9]$

Question 12 - Writing Nested Loops

```
1*2*3*4*5*6*7*8*9*
3*4*5*6*7*8*9*
5*6*7*8*9*
7*8*9*
9*
```

```
for (long i = 1; i <= 9; i += 2) {
    for (long j = i; i <= 9; j += 1) {
        cs1010_print_long(j);
        cs1010_print_string("*");
    }
    cs1010_println_string("");
}
```

Question 13 - Linear Recursion

Write a **recursive function** PrintBinary that prints the *binary representation* of a given positive integer x. For example,

- PrintBinary(3) prints 11
- PrintBinary(9) prints 1001
- PrintBinary(12) prints 1100

The function should have the header `void printBinary(long x)`, and can only call itself or CS1010 I/O library functions.

Question 13 - Linear Recursion

- Let's play with the problem a bit.
- Let's try what we are already familiar with: stripping the last digit at each recursive call and doing something with it
  - $n = 204$ is 1001100 in binary
  - `204 -> 4 -> ???`
  - `20  -> 0 -> ???`
  - `2   -> 2 -> ???`
- We end up only making 3 recursive calls.
- If we print out one digit per recursive call, we won't print out the required 7 needed for the binary representation
- For $n > 2$, the number of digits in base-10 is usually less than the base-2 representation

Question 13 - Linear Recursion

- Let's leverage something we know about the binary representation
- We can express any base-10 number as a sum of powers of 2
- For example,

$$12 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$

- **What about repeated divison by 2?** (integer division)
  - 12_{10} in binary is 1100
  - $12 \div 2 = 6$ with remainder 0
  - $6 \div 2 = 3$ with remainder 0
  - $3 \div 2 = 1$ with remainder 1
  - $1 \div 2 = 0$ with remainder 1
- See the pattern? When we do a division by 2, check the remainder r, if $r = 0$, print 1, otherwise print 0
- *This is our algorithm*

Question 13 - Linear Recursion

```
void printBinary(long n)
{
    if (n == 0) {
        return;
    }

    printBinary(n / 2);

    if (n % 2 == 1) {
        putchar('1')
    } else {
        putchar('0')
    }
}
```

Assignment 3 Comments

Q1: ID

- Generally done well
- We can store the characters concisely i.e `char codes[] = "YXWURNMLJHEAB"`

Q2: Kendall

- The most optimal solution is to do a double `for`-loop

```
long counter = 0;
for (long i = 0; i < n; i += 1) {
    for (long j = i + 1; j < n; j += 1) {
        if (a[i] > a[j]) {
            counter += 1;
        }
    }
}
return counter;
```

- (Optional): There's a way to count inversions in $O(n \log n)$ using a variant of MergeSort. Google for further reading!

Q3: Max

- For some codes, the base case was unnecessarily complex
- A simple `start >= end` suffices

```
long max(long a[], long start, long end) {
    if (start >= end) {
        return a[start];
    }
    long mid = (start + end) / 2;
    long m1 = max(a, start, mid);
    long m2 = max(a, mid + 1, end);
    return m1 > m2 ? m1 : m2;
}
```

Q4: Counting Sort

- This questions is focused on *efficiency*
- You want to take as few steps as possible to print the output

Some common mistakes:

- Looping over the input array 10001 times (You can do it in n steps)
- Building the frequency array in $O(n)$, then looping over the frequency array n times

Q4: Counting Sort

```
void counting_sort(long A[], long n)
    long F[10001] = {0};

    for (long i = 0; i < n; i += 1)
        F[A[i]] += 1

    for (long i = 0; i <= 10000; i += 1)
        if (F[i] > 0)
            print i, F[i]

    for (long i = 0; i <= 10000; i += 1)
        for (long j = 0; j < F[i]; j += 1)
            print i
```

Assignment 4 Tips

Q1: Selection Sort

SELECTIONSORT is an $O(n^2)$ sorting algorithm that sorts by using the following strategy

1. Let $i \leftarrow n$
2. Consider the array from $A[1..i]$
3. Scan the array for the maximum element, note its index
4. Swap the maximum element with the element at index i
5. Let $i \leftarrow i - 1$
6. If $i = 1$, terminate, otherwise, go back to Step 2

The algorithm is simple! But do take note of **negative array values**.

Q2: Mastermind

In this question, you play the game MASTERMIND for **eight** rounds.

Hints:

- The question is deceptively easy. You only get two test cases so make sure to test thoroughly
- *You need more arrays than just the two input arrays to do this question*