Assignment 1

COMP 206

Instructions

Submit a single pdf file named yourFirstName.pdf

There are 9 questions. Q1-Q8 are for 10 marks each, and Q9 is for 20 marks. All the programs must be written in C_0 .

---

1. Provide a one line answer for the following:
   - (a) What is the type of an array of boolean values? bool[]
   - (b) Declare an int array of size 100 and store it in a variable named `x` .
   - (c) What is the use of `-d` flag in the `coin -d` command? enables contracts.
   - (d) What is the type of a function with no return value? void
   - (e) Enumerate all the constants of type `bool` . true/false
2. Write a function named `abs` which takes an int and returns its absolute value.

$$\text{abs(x)} = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

3. Write a function named `log` , which is defined as follows for int values:

$$\log(\text{x}) = \begin{cases} 0 & \text{if } x = 0 \\ 1 + \log(\text{x}/2) & \text{if } x \geq 1 \\ ??? & \text{undefined otherwise} \end{cases}$$

   The function should have a precondition contract to handle the undefined case. Note that the division operator used above is the same as C_0 division.
4. Write a function named max2, which takes two positive ints and returns their maximum. Write a precondition to make sure that the caller supplied positive numbers. Write a postcondition for the fact that when x is the maximum of two numbers a and b, then we have $x \geq a$ and $x \geq b$.
5. Write a postcondition for the following function which adds two numbers. The postcondition should check that the returned value is `a+b` .

2.
```
int abs(int x){
    if (x >= 0){
        return x;
    }
    else {
        return -x;
    }
}
```

3.
```
int log(int x)
//@requires x >= 0;
{ if (x == 0 ){
    return 0;
  }
  else {
    return 1+log(x/2);
  }
}
```

47

```
int max2(int a, int b)
//@requires a>0 && b>0;
//@ensures \result >= a &&
          \result >= b;
//@ensures \result == a
          \result == b;
{
    if (a > b) {
        return a;
    }
    else {
        return b;
    }
}
```

Logical OR.

5)

```
int add(int a, int b)
//@ensures \result == a+b;
{
return a+b;
}
```

6) put extra parameter in funct :-

```
bool binarySearch(int[] a,
                  int item,
                  int len)
```

=> and use the foll :-

false <-> false

not found <-> !found

```
int add(int a, int b) {
    a = a+b;
    return a;
}
```

6. Convert the following python function (line by line) to search an integer array in C_0 . As there is no len function in C_0 to find the length of an array, write a function which takes length of the array also as an extra argument.

```
def binarySearch(alist, item):
"""
alist: list to search in
item:  element to search for
"""
        first = 0
        last = len(alist)-1
        found = False

        while first<=last and not found:
            mid = (first + last)//2
            if alist[mid] == item:
                found = True
            else:
                if item < alist[mid]:
                    last = mid-1
                else:
                    first = mid+1

        return found
```

7. Write a proof for the loop invariant $b^e r = x^y$ when e is an even number as discussed in the class on 27th August 2020. You can collaborate with classmates for this question only, but write your own proof after discussion (copying is not allowed). Also mention the names of your classmates you discussed with.

8. (a) Rewrite the following code without using the `while` keyword by converting it into a `for` loop.

```
int x = 1;
int t = 0;
while (x<10) {
    t = t + x;
    x = x+1;
}
```

(b) Rewrite the following code without using the `for` keyword by converting it into a `while` loop.

7) Modify the proof in notes, about odd nos., shared on canvas.

8) The question is about converting for ↔ while loops.

```
for (s1; s2; s3){
    body;
}
```

use this idea to convert while ⟶ for.

```
s1;
while (s2){
    body;
    s3;
}
```

```
int x;
for (x=1; x<10 && x*x<5; x=x+1) {
    x = x+1;
}
```

9. The function `sumInt` takes an int n as input and returns the sum of ints from 1 to $n-1$. You are already provided with the pre and postconditions along with the loop invariants. By using the code provided to you answer the following.

   - (a) Explain informally why the loop terminates.
   - (b) **INIT Step**: Prove that each loop invariant is true immediately before the loop condition is tested for the first time.
   - (c) **PRES Step**: Prove that if each loop invariant is true at the start of a loop iteration, then the loop invariants are also all true at the end of that iteration.
   - (d) Show that if the loop terminates, the postcondition must hold.

```
int sumInt(int n)
//@requires n > 0;
//@ensures 2 * \result == n * (n - 1);
{
    int sum = 0;
    int i = 0;
    while (i < n)
    //@loop_invariant 0 <= i && i <= n;
    //@loop_invariant 2 * sum == i * (i - 1);
    {
        sum = sum + i;
        i = i + 1;
    }
    return sum;
}
```

C1 C2 L3 L4 LG LI1 LI2 L8 L9 L10

9) a)

(1) $i = 0$ by L4.

(2) i is strictly increasing by LG.

(3) LG fails when $i >= n$

Combining ①, ②, ③ and noticing that a strictly increasing int sequence is finite when it is upper bounded.

9b) INIT :-

LI 1 :

T.P. $0 \leq i \leq n$ is true before LG is reached.

Proof:

① $i = 0$ by L4

② $0 \leq 0$ — fact

①+② $\Rightarrow 0 \leq i$, first half of ineq. proved.

③ $n > 0$ by C1.

①+③ $\Rightarrow n > i$

$\Rightarrow n >= i$, LI 1 proved.

LI2:-

$$2Sum = i'(i-1).$$

Proof:-

① $i' = 0$ by L4

② $sum = 0$ by L3

③ $2Sum = 0$ by ②

④ $i' * (i'-1) = 0$ by ①

by ③ + ④ we have

$$2Sum = i'(i-1)$$

hence proved.

9c) PRES step :-

LI1,

Assumptions
① Assume $0 \leq i' \leq n$ before the loop is executed.
② assume value of i as i^0 at the end of the loop.
③ notice n doesn't change in the loop body.
④ $i^0{}' = i^0 + 1$ by LG.
⑤ $i' > i$ using ④
⑥ ①+⑤ gives us $0 < i^0{}'$
⑦ $0 \leq i^0{}'$ using ⑥
⑧ $i < n$ by LG as we are going to execute the loop :
⑨ $i+1 \leq n$ using ⑧
⑩ $i' \leq n$ using ④
⑦+⑩ proves LI1.

LI2 : PRES step :

Assumptions
① Assume $2sum = i(i-1)$
② assume i, sum get changed to i', sum' by the end of the loop.

③ $i' = i + 1$ by L9

④ $sum' = sum + i$ by L8

⑤ $2sum' = 2sum + 2i$ using ④

⑥ $2sum' = i(i-1) + 2i$
using ① and ⑤

⑦ $2sum' = i(i-1+2)$ using ⑥
$= i(i+1)$
$= (i'-1)i'$ using ③

$\therefore 2sum' = i'(i'-1)$
Hence proved.

9d) The loop terminates when LG. is false.

① This gives us that $i < n$ is false ie. $i \geq n$.

② LI1 gives us that $i \leq n$

$\therefore \quad n \leq i \leq n$ by ①+②

$\Rightarrow \quad i = n$

③ Outside the loop at L10 we have $i = n$.

④ \result = sum by L10

⑤ By LI2 & ③ we have

sum = $i(i-1) = n(n-1)$

⑥ 2 \result = $n(n-1)$ by ④

$\therefore$ C2 holds hence proved.

Attention!!

Are we done?

Or does the picture change because of the fact that int is bounded by $2^{31}-1$?

The proofs above work well when int's are unbounded. Remember that overflow will occurs when nos. become larger than $2^{31}-1$.

Exercise :- ← assignment 2 ?

Can you find out the places where our proofs fail for 32 bit ints? Which of these places are fixable?

1. We may have a bug at each place the proofs fail.

Q. Debug the program!
(Hint: contracts???).