COMP 206 (T4 - 2020)

Final Hints and Solutions

Short Answer

1. An int is represented using the Two's **complement**.
2. An array is allocated using the function **alloc_array**.
3. A **NULL** pointer can not be dereferenced.
4. The `-d` flag enables **contracts**.
5. For any type T, the return type of alloc(T) is **T***.
6. An array of int is of type **int[]**.
7. For any type T, an array of T is of type **T[]**.
8. For any array of size n, the valid indices are from **0** to **n-1**.
9. What is the type of expression 1==2 ? Ans: **bool**
10. The function frames are stored in the **local** memory and the structs are stored in the **allocated** memory.
11. An int is of size **32** bits.
12. A variable var is declared using T var = alloc(struct abc). What is T? Ans: **struct abc***
13. An error is thrown if a variable is used without initializing it. Ans: **True**
14. The name of the C_0 interpreter is **coin**
15. The name of the C_0 compiler is **cc0**
16. According to pigeon hole principle the probability of two pigeons sharing birthdays out of a group of 26 pigeons is more than half. Ans: **False**
17. A **stack** is based on the last in first out principle.
18. What is the smallest number of nodes possible in a linked list with cycle? Ans: **One**
19. Suppose we have a hash table of size m. We know that there are n elements stored in it without collisions. What is the relation between n and m? **Ans:** $n \leq m$**, Using Pigeon Hole Principle.**
20. In a min-heap implemented using arrays, the index of the parent of a node stored at index k is `k/2`.

Answer the following about contracts.

1. Which contract is used to write the post-condition of a function? Ans: **ensures**

2. Which contract is used to write the pre-condition of a function? Ans: **requires**

3. If a function call is not safe then one of the **pre-condition** contract is not satisfied.

4. A function is said to be correct, if it satisfies all the **post-condition** contracts on a safe call.

5. Convert the contract //@requires x==1; into a multiline contract. Ans: `/*@requires x==1; @*/`

Write a function, which takes two positive ints and returns their minimum. Write a precondition to make sure that the caller supplied numbers strictly greater than zero. Write a postcondition for the function using the fact that, when x is the minimum of two numbers a and b, then we have a >= x and b >= x.

Hint: Refer to assignments

Answer the following about the function sumInt:

1. Is sumInt(-13) a safe call? Why? Ans: **No**
2. How many times is line `i=i+1;` executed on a function call of sumInt(5)? Ans: **Five (n times)**
3. What is the value returned by sumInt(2)? Ans: **One**
4. What is the value of i just before line return sum; is executed? Ans: **n**
5. Where does the loop-invariant `2 * sum == i * (i - 1)` fail in the code? Why?

   Ans: **By definition, loop invariant does not fail anywhere before the loop body is executed each time or after it is exited. Hence, it may fail somewhere in the middle of the loop. Here it fails just after line 11 is executed. It does not fail because of int overflow due to modular arithmetic. Even after overflow, both the LHS and the RHS are still equal.**

```
 1  int sumInt(int n)
 2  //@requires n > 0;
 3  //@ensures 2 * \result == n * (n - 1);
 4  {
 5      int sum = 0;
 6      int i = 0;
 7      while (i < n)
 8      //@loop_invariant 0 <= i && i <= n;
 9      //@loop_invariant 2 * sum == i * (i - 1);
10      {
11          sum = sum + i;
12          i = i + 1;
13      }
14      return sum;
15      int ans = 0;
16      return ans;
17  }
```

Long Answers

LA1

The function terminator does not terminate for all the input values.

Discuss the termination conditions for the while loop contained in it.

```
void terminator(int lo, int hi)
//@requires 0 <= lo && lo <= hi;
{
    while( lo != hi ) {
        lo = lo + 1;
        hi = hi - 1;
    }
}
```

Ans: The difference between lo and hi decreases by 2 each time the loop is executed. THe loop terminates when `lo==hi` or `lo-hi==0`. If the starting difference `hi-lo` is even then only can it reach zero by subtracting twos. This gives us the terminating condition:

1. If `lo-hi` is even the loop terminates.
2. If `lo-hi` is odd we get an endless loop.

LA 2

Prove the assertion, assuming the loop invariant.

```
void scanIntervalReverse(int lo, int hi)
//@requires 0 <= lo && lo <= hi;
{
    int i;
    for ( i = hi-1; i >= lo; i-- )
    //@loop_invariant lo-1 <= i && i < hi;
    {
        //body
    }
    //@assert i==lo-1;
}
```

Hint: Loop guard false and loop invariant true.

LA3

Prove the loop invariant.

```
void evens() {
    int i=0;
    while( i < 10 )
    //@loop_invariant i%2 == 0;
    {
        i = i + 2;
    }
}
```

Hint: For the PRES step `i'= (i + 2)%2 = i%2`

LA 4

Prove the correctness of the following function, assuming the loop invariant.

```
int nextEvens()
//@ensures \result % 2 == 0;
//@ensures \result == 10;
{
    int i=0;
    while( i < 9 )
    //@loop_invariant i%2 == 0;
    {
        i = i + 2;
    }
    return i;
}
```

Hint: For second ensures show that loop exits with i=10, and the first ensures follows from LI.

LA 5

A binary search tree is called as zig zag if it has the shape as shown in the figure.

1. What is the order in which you would insert 1,2,3,4,5 into an empty binary search tree so that you get a zig-zag tree? Justify your answer using the ordering property of the binary search trees. (No partial marks if reason is not provided). **Ans: The sequence of inserts starting with 1 is 1, 5, 2, 4, 3. This is the only possible answer.**

2. Generalize your observation to find the order of inserting first n natural numbers 1, 2, ..., n to obtain a

zig-zag tree. [Hint: Which number is at the root node? Which number is the right child of root?] **Ans: The justification follows from ordering property. The ordering property implies the root to be the min element and its child to be the max element. The rest of the elements follow from the recursive structure of the zig-zag tree.**

LA 6

What does the function whoami in the following C code do? Justify your answer.

```c
#include <stdlib.h>
#define N 1

int whoami(int a, int b, int c) {
    int* x = malloc( N * sizeof( int ) );
    *x = a;
    if( b > *x ) {
        *x = b;
    }
    if( c > *x ) {
        *x = c;
    }
    return *x;
}
```

Hint: max of a, b, c.

LA 7

You are the organizer of a knockout boxing tournament. There are $n = 2^k$ players participating in the tournament, for some positive int k. Each round consists of randomly pairing players and holding a match between them. Only the winners will proceed to the next round (there are no draws). When only one player remains the tournament stops. For example with n=2 players, the tournament will have only one round. For n=4, lets say P1, P2, P3 and P4 are the players. Lets say the first round consists of a match P1 vs P2, and P3 vs P4. If P1 and P3 are winners in the first round then the tournament will finish in 2 rounds after a match of P1 vs P3.

1. What is the number of rounds for any n=2^k ? **Ans: In each round the number of players gets halved. Therefore, the number of rounds is $\log_2(n) = k$.**
2. How many matches does the winner play? **Ans: Winner plays each round, hence the winner plays k matches.**
3. How many matches happen in total? **Ans: n-1. Hint: How many matches happen in the each round?**