Quiz #3 Solutions

Name: _____

Student Number: _____

Signature: _____

Instructions

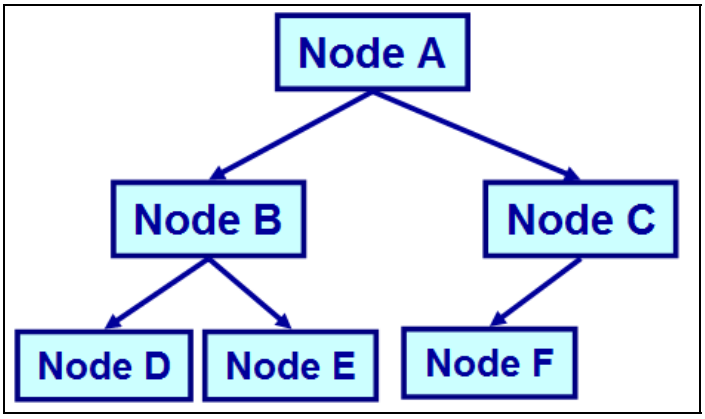
1. **Fill in your Name, Student Number, and signature above.**
2. **This is a closed book Quiz. No electronic or paper aids permitted.**
3. **Do not open this test booklet until instructed to do so.**
4. **Clearly indicate if some part of your work is not to be marked. Add as many comments as needed to provide a clear response.**
5. **You may answer the questions in any order you want.**
6. **Raise your hand if you have a question. The instructor will come over to assist you.**
7. **Copying from or communicating with a neighbor or with anyone directly or electronically will result in both students receiving a zero and may result in further disciplinary action by the school and or university administration.**
8. **The total number of points for this Quiz is 50.**
9. **You may use the attached Operator Precedence chart and Syntax chart**
10. **You will have 20 minutes to complete this Quiz.**
11. **When you are finished, bring your paper and student card to the front of the room where you will hand in your quiz.**

Good luck!

Total = _____ / 50

Question	Max Mark	Actual Mark
1	5	
2	5	
3	10	
4	10	
5	10	
6	10	
Total	50	

1. Answer the following questions about the tree below. 5 Marks



a. **Identify the Root of this tree:** A

b. **List the Ancestors of Node E:** B and A

c. **What is the Height of the Tree?** 2

d. **Is the tree Complete?** Yes

e. **What is the Order of this tree?** 2 (It is a Binary tree)

2. Complexity 5 Marks

What is the complexity of the following code fragment using Big-O notation with respect to the value of n?

```
for (int count1 = 0; count1 < n;   count1++)
for (int count2 = 0; count2 < 2*n; count2++)
   cout << count2*count2*count2 << endl;
```

- ☐ n + 2n
- ☐ O(n + 2n)
- ☐ $O(n^3)$
- ☒ $O(n^2)$
- ☐ O(1)
- ☐ None of the above

3. Recursive Functions - fibonacci 10 Marks

The fibonacci sequence is a famous bit of mathematics, and it happens to have a recursive definition. The first two values in the sequence are 0 and 1 (essentially 2 base cases). Each subsequent value is the sum of the previous two values, so the whole sequence is: 0, 1, 1, 2, 3, 5, 8, 13, 21 and so on. Define a recursive fibonacci(n) method that returns the nth fibonacci number, with n=0 representing the start of the sequence.

Your function must have the following signature:

```
int fibonacci(int n);
```

For example:

```
fibonacci(0) → 0
fibonacci(1) → 1
fibonacci(2) → 1
fibonacci(3) → 2
fibonacci(4) → 3
```

```
int fibonacci(int n) {

      if (n == 0)
           return 0;
       else if (n == 1)
           return 1;
       else
           return fibonacci(n-1)+fibonacci(n-2);
```

4. Dynamic Arrays as Return Values - tripleChar 10 Marks

Given a C string parameter called `str`, return a C string where for every char in the original, there are three chars. Your function will accept a nullptr or a C string that is terminated by a null (zero) character, and must return a new dynamic char array that is also terminated by a null character, and that contains three characters for every character in the `str` tripleChar.

Your function must have the following signature:

```
char* tripleChar(const char str[]);
```

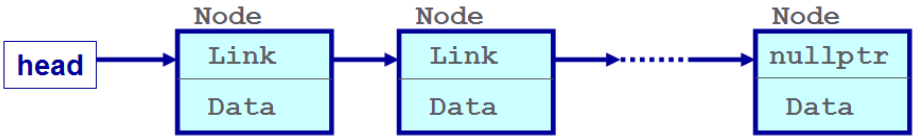
For example:

```
tripleChar("The")        → "TTThhheee"
tripleChar("AAbb")       → "AAAAAAbbbbbb"
tripleChar("Hi-There")   → "HHHiii---TTThhheeerrreee"
tripleChar(nullptr)      → ""
```

```
char* tripleChar(const char str[]) {

  int len = 0;

  if (name != nullptr)
      len = strlen(name);

  char* retStr = new char[3*len+1];

  for(int i=0; i<len; i++) {
     retStr[3*i  ]=name[i];
     retStr[3*i+1]=name[i];
     retStr[3*i+2]=name[i];
  }

  // Add null char at the end to terminate the C string
  retStr[3*len] = 0;

  return retStr;
```

5. Implement Linked List functions below using `struct` Node 10 Marks

```
struct Node {
    Node    *link;
    int      data;
};
typedef Node* NodePtr;
NodePtr head = nullptr;
```



(a) Complete the head_insert routine below: 5 Marks

```
/*****************************************************************************
 * Creates a new Node, sets is data field to the_number, and adds it as the
 * first node pointed to be head.
 *****************************************************************************/
 void head_insert(NodePtr& head, int the_number) {

 NodePtr newNode = new Node;
 newNode->data   = the_number;
 newNode->link   = head;

 head = newNode;



}
```

(b) Complete the search routine below: 5 Marks

```
/*****************************************************************************
 * This function searches the linked list for a node with data == dataToFind
 * and returns a pointer to the node if found, and returns nullptr otherwise.
 *****************************************************************************/
NodePtr search(NodePtr head, int dataToFind) {

NodePtr here = head;

while (here!=nullptr && here->data != dataToFind)
     here = here->link;

return here;

}
```

6. Polymorphism: What output does main() produce? 15 Marks

```
class Animal {
public:
    Animal(string animalType, string sound);
    virtual void speak();
    string animalType;
    string sound;
};

Animal::Animal(string animalType, string sound) {
    this->animalType = animalType;
    this->sound      = sound;
}

class Bee    : public Animal{ public: Bee (); };
class Lion   : public Animal{ public: Lion(); void speak() override;};
class MtnLion: public Lion  { public:         void speak() override;};

void Animal ::speak(){cout <<animalType <<"s say "<< sound << endl;}
void Lion   ::speak(){cout <<"Big Cats ROAR!"              << endl;}
void MtnLion::speak(){cout <<animalType <<"s purr"         << endl;}

Bee ::Bee() : Animal("Bee",  "Buzz Buzz") {}
Lion::Lion(): Animal("Lion", "Meow"     ) {}

int main(int argc, char** argv) {
    Animal  bear("Bear", "Grrrr!");
    Animal* zoo[4] = {&bear, new Bee, new Lion, new MtnLion};
    Animal  lion   = *zoo[3];

    cout << "Zookeeper, Zookeeper, What do you hear?\n";
    for(Animal* animal:zoo    )  animal->speak();
    lion.speak();
}
```

(write the output produced by executing main here)

```
Zookeeper, Zookeeper, What do you hear?
Bears say Grrrr!
Bees say Buzz Buzz
Big Cats ROAR!
Lions purr
Lions say Meow
```