SPRING SEMESTER 2004

CMSC 246

ADVANCED PROGRAMMING IN C++

FINAL EXAM

READ THE FOLLOWING BEFORE YOU START!!

- Make sure to write your name and SSN on the top of every page in this exam book. We do not separate problem and answer sheets in this exam
- This exam is closed-book and closed-notes.
- You are required to answer all questions.
- Please write all answers in the exam book.
- Please also manage your time well since you only have 90 minutes to finish the exam.
- GOOD LUCK!!
1. (20 points) Given the following node definition of the binary tree node:

```cpp
class BTreeNode {
    public:
        BTreeNode * left;
        BTreeNode * right;
        int val;
    }
```

(a) Complete the partially implemented class BTree to implement the binary tree operations. Specifically, you need to implement the `insert` and `search` methods.

```cpp
class BTree {
    public:
        BTree()
        {
            root=NULL;
        }
        boolean insert(BTreeNode * newnode);
        BTreeNode * search(int val);

    private:
        BTreeNode * root;
    }
```

(b) Starting with an empty tree, a sequence of number is inserted in the following order: {8, 12, 3, 5, 9, 26, 11, 3, 6, 7, 4}; what will be the resultant tree? You can use graphical representation by drawing the final tree.

(c) What is a balanced binary-search-tree? what is the search cost for a balanced binary-search-tree?

(d) Implement the post-order traversal algorithm. If you apply the post-order algorithm to the tree in (b), what is the output (or visiting order of the nodes)?
NAME: 

SSN: 

2. (20 points) The following is the partition algorithm used in the quicksort.

```c
int partition(int * a, int left, int right) {
    int i=left-1, j=right;
    int pivot=a[right];
    for (; ;) {
        while (a[++i] < pivot);
        while (pivot<a[--j])
            if (j==left) break;
            else;
        if (i>=j)break;
        exch(a[i], a[j]);
    }
    exch(a[i], a[right]);
    return i;
}
```

In this implementation, the pivot value is selected as the rightmost array member.

(a) Given an array `a[6]={5,5,5,5,5,5}` , how many exch() function calls will be executed during the partition? The input parameters for the partition call are: `left=0`, and `right=5`.

(b) if we change line 5 to `while(a[++i]<=pivot);`, will the algorithm still work correctly?

(c) revise the partition algorithm such that the pivot value is selected as the leftmost array member.
NAME:  

SSN:
3. (15 points) answer the following questions related to hash table.

(a) What is the functionality of hash function?
(b) What is the collision in hash table?
(c) what is the cost of insert operation in hast table?
(d) what is the cost of the search operation in hash table?
4. (15 points)

(a) what is the purpose/result of the following code segment?

```cpp
int array[6]={1,2,3,4,5,6};
std::vector<int> integers(array, array+6);
std::ostream_iterator<int> output( cout, "*" );
std::copy( integers.begin()+1, integers.end(), output);
```

(b) How is exception handled in C++? What is the output of the following program?

```cpp
void throwException()
{
    try {
        cout << " Function throwException throws an exception\n";
        throw exception(); // generate exception
    } catch ( exception &caughtException ){
        cout << " Exception handled in function throwException" 
             << "\n Function throwException rethrows exception\n";
        throw; // rethrow exception for further processing
    }
}

int main()
{
    try {
        cout << "\n main invokes function throwException\n";
        throwException();
        cout << "This might not print\n";
    } catch ( exception &caughtException ){
        cout << "\n Exception handled in main\n";
    }
    cout << "Program control continues after catch in main\n";
}
```

(c) What is the rethrowing in exception handling in C++? Why do we need it?

(d) What is the effect of throwing an exception in your code without using try and catch?
5. (15 points) The following code is intended to replace all occurrence of substring `sub1` by a new string `nst` in the original string `str`:

```c
void replacestr(char * str, char * sub1, char * nst)
{
    int lsub1= strlen(sub1);
    int lstr=_strlen(nst);
    int location = findfirst(str, sub1);
    if (location!=-1) {
        movestr(&str[location+lsub1],&str[location+lstr]);
        replacestr(&str[location+lsub1],sub1, nst);
    }
}
```

Here the `strlen` return the length of the char string, and `findfirst` return the location of first occurrence of `sub1` in `str`. `findfirst` return -1 if `sub1` is not found in `str`. `movestr (char * src, char * dest` move the substring at `src` to `dst`.

Will the above implementation generates the desired result? If not, what is the problem and how to fix it?
6. (15 points):
   (a) What is the independent inheritance?
   (b) what is the virtual inheritance?
   (c) what is the output of the following code?

```cpp
#include<iostream> using std::cout;
class A { public:
    int y;
    virtual void foo1(){};
};
class C : virtual public A
{
    public:
        int x;
};
class B : virtual public A {
    public :int x;
    virtual void foo1() {cout<<"I’m B::foo1\n";}
};
class D : public B, public C {};
main(){ C c; B b; D d;
c.y=1;
b.y=2; cout<<c.y<<endl; cout<<b.y<<endl;
d.B::y=3; d.C::y=4;
cout <<d.y<<endl;
cout <<d.B::y<<endl;
cout <<d.C::y<<endl;
}
```