Corrections

2) You can instantiate an abstract class. False (Page 9)

5) Recursive functions must always return a value. False (Page 75)

10) By default, arrays and structures are passed by reference. False (Page )

11) What is specification? (Page 33)

An operational contract between the developer and the end user.

13) A reason recursion can be inefficient is: The overhead associated with function calls. (Page 75)

20) The two parts to a recursive function are:

(a) a stopping condition and (b) function recall. (Page 76)

21) A static object is an object that keeps its value after the function termination and reuses that previously held value when the function is called upon again in the future. (Page 72)

22) The copy constructor gets implicitly invoked when you (Page 60)

(a) Declaration and initialization in one step

(b) Passed by value

(c) Returned by value

23) Consider the following function:

Void RecEm( int i )

{

If ( i < 8 ){

i++;

RecEm(i);

cout << i << ‘ ‘;

}

}

What is produced when RecEm is called with a parameter value of 4? (Page 81)

8 7 4 5

24) Given the following function,

Void ShowTosses( int coinCount)

{

If ( coinCount!=0 ){

Cout << ‘H’;

ShowTosses(coinCount – 1);

cout << ‘T’;

ShowTosses(coinCount – 1);

}

}

What is the exact output produced by the function call ShowTosses(2) (Page 81)

HHTTHT

31) Node::Node(newData, nextNode)

{

data = newData;

next = nextNode;

}

32) void list::printNode(cons int location) const

{

Node\* temp = head;

Int ndx=0;

While (ndx<location)

{

temp = temp -> next;

ndx++;

}

if (temp!=NULL && ndx == location)

cout << temp->data;

}

33) list& list:: operator = (const list& source)

{

Node\* temp = head;

Node\* stemp = source.head;

bool equality = true;

while (equality == true)

{

if (temp->data!=stemp->data)

{

equality = false;

}

temp = temp->next;

stemp = stemp->next;

}

if (temp == NULL && stemp == NULL && equality == true)

return &this;

clear();

if (!source.isEmpty())

{

stemp = source.head;

head = new Node (stemp->data, NULL);

temp = head;

while (stemp-> != NULL)

{

stemp = stemp -> next;

temp ->next = new Node (stemp->data,NULL);

temp=temp->next;

}

}

return &this;

}