

Spring 2013: CS 3323 Midterm Exam 1

Total Time: 40 minutes

Total Points: 45

Write your name clearly. Answer all the questions.

Reead all the questions. If you need more space, use the other side of your question sheet.

Name: _____

Date: _____

Question 1. True/False

[10]

- true 1. Garbage collection is a form of memory management.
- false 2. Deleted dynamic elements make garbage collection a tougher task.
- true 3. List can grow indefinitely.
- true 4. Array based implementation of list may require shifting of elements when elements are added/deleted.
- false 5. List is a LIFO based data structure.
- false 6. By default array data structure has inbuilt out of bound error checking.
- true 7. Node element in linked list implementation has members for storing value and storing addresses.
- false 8. Stack is not a good choice for conversion of infix operation into postfix operation.
- false 9. Deep copy constructor should not be written for dynamic elements.
- true 10. Elements are added and deleted at the end (top) of the stack.

Question 3. Analyzing code segment.

[12]

Observe the following declarations:

```
class Node{ public:
    int data;
    Node * next
}; Node *p1 = new Node, *p2 = new Node, *p3 = new Node;
```

Tell what will be displayed:

```
p1->data = 12; p2->data = 34;
p1->next = p2;      p2->next = p1;
cout << p2->data<< " " << p2->next->data<<endl;
cout << P1->next->data<< " " << P1->next->next->data;
```

34 12
34 12

<pre>P1->data = 12; p2->data = 34; *P1 = *P2; cout << p2->data<< " " << p2->next->data<<endl; cout << P1->next->data<< " " << P1->next->next->data;</pre>	nothing
<pre>P1->data = 12; p2->data = 34; p3->data = 34; P1->next = p2; P2->next = p3; P3->next = 0; cout << P1->data<< " " << p1->next->data<< " " << p1->next->next->data << " "<<p3->data;</pre>	12 34 34 34

For next set of problems, assume that `Stack` is the class implemented by using static arrays and can hold integer values (the one we discussed in the class). The capacity of stack is set to 5. **Give the value of `myTop` and the contents of the array referred to by `myArray` in the stack `s` after the code segment is executed, or indicate why an error occurs.**

<pre>Stack s; s.push(123); s.push(456); s.pop(); s.push(789); s.pop(); s.pop();</pre>	<pre>myTop= 0; myArray [0] = 123, other elements are undefined</pre>
<pre>Stack s; s.push(222); int i = s.top(); s.pop(); s.push(i); s.pop();</pre>	<pre>myTop = 1; myArray[0] =222; other elements are undefined</pre>
<pre>Stack s; for(int i = 1; i < 5; i++) s.push(i*i); s.pop();</pre>	<pre>myTop = 2; myArray[0]= 1, myArray[1] = 4, myArray[2] = 9, myArray[3] = 16, myArray[4] =25</pre>

Question 4. Explain the following concepts:**[6]****a. Stack-ADT:**

A sequence of data items that can be accessed at only one end, called the top of the stack.

Basic operations are: construct an empty stack, check if stack is empty, push a value onto the top of the stack, retrieve the top value, and remove the top value.

b. List-ADT:

A list is an abstract data type that describes a linear collection of data items in some order, in that each element occupies a specific position in the list. The order could be alphabetic or numeric or it could just be the order in which the list elements have been added. Unlike a set, the elements of a list do not need to be unique.

c. Role of Activation Record in recursive function calls:

An activation record is a private block of memory associated with an invocation of a procedure. It is a runtime structure used to manage a procedure call. An AR is used to map a set of arguments, or parameters, from the caller's name space to the callee's name space.

An AR includes a mechanism to return control to the caller and continue execution at the point immediately after the call.

Question 5. Convert the following infix expressions into postfix expressions**[5]**

a. $(a + b) * (c * (d - e) / f)$

$ab + cde - *f / *$

b. $(7 * 8 - (2 + 3)) \% 2 + 2$

$78 * 23 + - 2 + \%$

Question 4. Observe following declaration and description for a list:**[3+3]**

```
class list {
class Node{
public:
    datatype value;
    Node *next;
    Node():next(0){}
    Node(datatype val):value(val),next(0){}
    } ;
```

```
Node *first;
```

```
int mySize;
```

```
public:
```

```
~List();
```

```
/*-----
```

```
Destructor
```

```
Precondition: This list's lifetime is over.
```

```
Postcondition: This list has been destroyed.
```

```
-----*/
```

```
int NodeCount();
/*-----
NodeCount: Counts the number of nodes in List object
Precondition: None.
Postcondition: None.
-----*/};
```

Following concepts described in the class, provide a destructor and and a Nodecount function that returns numbers of nodes in List for this list class.

Question 4. Observe following declaration for a Stack:**[3+3]**

```
class Stack {
public:
    /***** Function Members *****/
    /***** Constructors *****/
    Stack();

    Stack(const Stack & original);
    /*-----
       Copy Constructor
    -----*/
    /***** Destructor *****/
    ~Stack();

    void pop();
    /*-----
       Remove value at top of stack (if any).

       Precondition: Stack is nonempty.
       Postcondition: Value at top of stack has been removed,
    -----*/
    void push(int value);
    /*-----
       Add the value at top of stack
       Precondition: None
       Postcondition: Value has been added at top of stack.
    -----*/

private:
    /*** Node class ***/
    class Node
    {
    public:
        int data;
        Node * next;
        /**--- Node constructor
        Node(StackElement value, Node * link = 0)
        { data = value; next = link; }

    };

    /***** Data Members *****/
    NodePointer myTop;    // pointer to top of stack
};
```

Following concepts described in the class, provide push and pop methods

[Bonus Question] Observe following declaration for a polynomial class

[6]

```
class Poly {  
    int Degree;  
    int Cofact[capacity]  
public:  
    /**** Function Members *****/  
  
    /**** Constructor *****/  
  
};
```

With the above definitions, provide an overloaded operator for addition operation, in other words overload '+' operator for this class.