	MIDTERM EXAMINATION SEMESTER SPRING 2004 CS301-DATA STRUCTURE	Total Marks:75 Duration: 60min

Instructions

Please read the following instructions carefully before attempting any question:

1. **The duration of this examination is 60 Mins.**
2. **This examination is closed book, closed notes, closed neighbors; any one found cheating will get no grade.**
3. **Unless stated otherwise, all questions carry a single mark.**
4. **Do not ask any questions about the contents of this examination from anyone.**
   a. **If you think that there is something wrong with any of the questions, attempt it to the best of your understanding.**
   b. **If you believe that some essential piece of information is missing, make an appropriate assumption and use it to solve the problem.**
5. **You are allowed to use any development environment like Dev C++ etc.**

Question No: 1 **Marks: 5**

What is the value of the C/C++ expression: 6.2 + 5 / 3

Answer:

6.2+5/3

6.2+1 **(As 5 & 3 are integers so result will be in integer, fraction portion will be discard)**

7.2

Question No: 2 **Marks: 5**

What is the value of the postfix expression: 8 6 4 + 3 * + 5 +

Answer:

Input	**op1**	**op2**	**value**	**stack**
8				8
6				6 8
4				4 6 8
+	6	4	10	10 8
3				3 10 8
*	10	3	30	30 8
+	8	30	38	38
5				5 38
+	38	5	43	43

Value of Postfix Expression = 43

Question No: 3 **Marks: 5**

Write an equivalent postfix expression for the infix expression:

6 + (5 - 3) * 8

assuming standard precedence for order of operations.

Answer: **6 5 3 - 8 * +**

Step No.	Symbol	Postfix	Stack
1	6		
2	+	6	+
3	(	6	+(

4	5	6 5	+(
5	-	6 5	+(-
6	3	6 5 3	+(-
7	)	6 5 3 -	+
8	*	6 5 3 -	+*
9	8	6 5 3 - 8	+*
10		6 5 3 - 8 * +	

Question No: 4 **Marks: 10**

A linked list node class is declared as follow:

```
class Node
{

public:
        Node(const string& s, Node* ptr)
                    : info(s), next(ptr)
        { }
private:
        string info;
        Node* next;
};
```

Write a function that changes every ’t’ that occurs as the first letter of a word to a ’b’. No other letters should change. For example, ("tin", "tile", "ant", "saint", "tot") should be changed to ("bin", "bile", "ant", "saint", "bot")

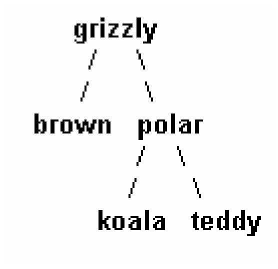
```
void change(Node* list)
// post condition: all t’s that occur as first letters of a
// word in the list nodes are changed to b’s
{
```

Answer:

```
Void change(Node * list)
{
  temp = list;
  While(temp !=NULL)
  {
     If (temp->info[0]==’t’)
        Info[0]=’b’;
     Temp=temp->next;
  }  //end of while
 }
```

Question No: 5 **Marks: 15**

The tree below is a binary search tree.



a) What is the preorder traversal of the tree?

b) Add nodes containing “black” and “panda” so that the tree remains a search tree. Add “black” first. Draw the nodes attached to the tree diagram above.

c) Draw a search tree in which “teddy” is at the root of the tree, and the root’s left child is “polar” (include all other nodes from the tree in the diagram above, these other nodes can occur in any order in the tree you draw.)

Answer

a) grizzly, brown, polar, koala, teddy

b)
```
                    grizzl
                   /      \
              brown      polar
             /           /   \
         black      koala    teddy
                        \
                        panda
```

c)
```
                    teddy
                   /
               polar
              /
          grizzly
         /       \
     brown      koala
```

Question No: 6 **Marks: 10**

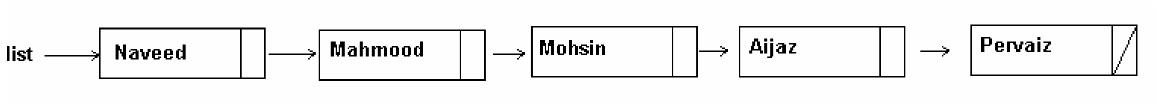
Draw a box and pointer diagram (i.e., a linked list) of what list points to after executing the following code fragment:

```
string* name;
Stack<string*> s;
Node* list = NULL;
s.push(new string("Naveed"));
s.push(new string("Mahmood"));
s.push(new string("Mohsin"));
s.push(new string("Aijaz"));
```

```
s.push(new string("Pervaiz"));

while (!s.empty())
{
        name = s.pop(name);
        list = new Node(name, list);
}
```

Answer:



Question No: 7 **Marks: 5**

Consider the following Mystery function.

```
int Mystery(int num)
{
        if (num > 4)
        {
                return 2 + Mystery(num - 5) + Mystery(num - 2);
        }
        return 1;
}
```

What is the value returned from the call Mystery(5)?

Answer:

4

Question No: 8 **Marks: 20**

In the syntax of most programming languages, there are some characters that occur only in nested pairs, which are called bracketing operators. C++ has the following bracketing operators:

```
( ::: )
[ ::: ]
{::: }
```

In a properly formed program, these characters will be properly nested and matched. To determine whether this condition holds for a particular program, you can ignore all the other characters and look

simply at the pattern formed by the parentheses, brackets, and braces. In a legal configuration, all the operators match up correctly, as shown in the following example:

{ x = (s = v[1] + 2); y = 4 * (v [v.size() - 1] + x); }

The following configurations are illegal for the reasons stated:

- **(([4])The line is missing a close parenthesis.**
- **AB) (The close parenthesis comes before the open parenthesis.**
- **{(x}) The parentheses and curly braces are improperly nested.**

For this problem, your task is to write a function

bool isBalanced(string s)

that takes a string s with all characters except the bracketing operators removed. For example, for the program statements

{ x = (s = v [1] + 2); y = 4 * (v [v.size() - 1] + x); }

The string s would contain

{ ([ ]) ([( )]) }

The method should return true if the bracketing operators in s are balanced, which means they are correctly nested and aligned, otherwise it should return false. You must either use a Stack or recursion in your solution. Assume you have the following helper functions.

bool IsOpener(char ch); // returns true if ch is ({ or [, else false

bool IsCloser(char ch); // returns true if ch is) } or], else false

char MatchingChar(char ch); // returns matching char.
// In other words, returns { for },
// } for {,] for [, and so on
// Returns if ch is not opener
// or closer

boolean isBalanced(String s)
{

Answer:

```
bool is Balanced(string s)
{
 char ch, stch;
 int i =0;
 while(s[i])              // untill the end of the string
 {
   ch = s[i];             // pick the character
   if( boolisOpener(ch);              // if it is opener then push on the stack
            st.push(ch);
  else if( boolisCloser(ch)        // if it is closer then pop the last char. from stack
    {          stch = st.pop();

             if( ch != MatchingChar(stch)     // match the current char with stack char
                            return false;                  // return false if not match
    }
    i++;                                                    // go to the next character
 }

if(st.isempty())                          // at the end stack should be empty
     return true;                         // if empty then return true
else
     return false;                    // else return false;
}
```