Department of Computer Science and Engineering
CS 101 : Computer Programming and Utilization
QUIZ 2, AUTUMN 2013– 2014 SESSION

Day and Date : Friday, 11/10/2013, HALL 1 to 4 **Marks : 20**
Time : 8:15 – 9:15 hrs **Weightage : 10%**

QUESTION PAPER AND ANSWER BOOKLET
INSTRUCTIONS

==

1. PLACE MOBILES AND ALL ELECTRONIC GADGETS, IN SWITCHED OFF STATE, IN YOUR BACKPACKS AND KEEP THEM INSIDE THE HALL AT THE ENTRANCE.

2. Answer to each question is to written in the space provided for the same. For rough work, use the blank / extra pages provided. Submit this booklet at the close of examination.

3. SINGLE HANDWRITTEN A4 SHEET, non-transferable, is permitted to be used during the examination.

4. Minimum penalty for use of unfair means is FR grade.

5. Answers without justification will not get full credit.

6. No clarifications on the questions will be given during the examination, When in doubt, make reasonable assumptions, state them clearly, and write your answer.

7. Check that this booklet has 12 pages and 3 questions. In case of mismatch ask for another booklet.

8. Write your Roll No and Lab batch on the front cover and your roll number at the top of every sheet.

==

ROLL No. : ____________________ **LAB BATCH : ________/ OSL | NSL**

NAME : ____

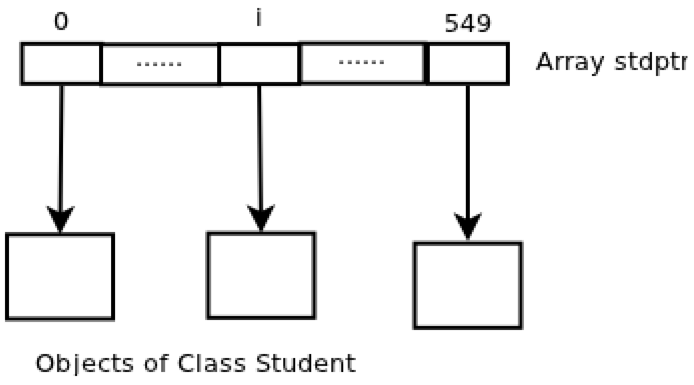
	Q1	**Q2**	**Q3**	**TOTAL**
Marks Allotted	**5**	**3**	**12**	**20**
Marks obtained				
Initials of Examiner / Scrutinizer				

Q1. Consider the data structure drawn below along with the details provided.

Attributes of data members of class **Student** are given below. For roll_no, you have to choose an appropriate type that is consistent with IITB norms, while mobile_no is given to be a 12 digit number.

Attribute No.	1	2	3	4	5	6
Attribute Name	First_name	last_name	roll_no	age	marks	mobile_no
Attribute Qualifier	public	public	public	private	private	private
Attribute Type	string	string		unsigned short	int array of size 5	

(a) Write the definition of the class Student that contains attributes as per the above table. Create an array of pointers, **stdprt** to objects of Student class for each of 550 students as shown in the figure below.



(b) Write a function total_marks() which will calculate the total marks of a student and return its value. Total marks is the total of all marks given in marks[] attribute in class Student. The intended usage of the function total_marks can be any one of the two alternatives given below

Student s;
cout << s.total_marks() << endl;
	Student s;
	cout << total_marks(s) << endl;

Write answer to both (a) and (b) in the same space provided below

ROLL NO : ________________

ANSWER :

```
class Student
{
public :
string First_name;
string last_name;
string roll_no;
unsigned short age;
string mobile_no;

//Alternative 1
int total_marks()
{
int total = 0;
for(int i=0;i<5;i++)
total = total + marks[i];
return total;
}

//Alternative 2
friend int total_marks(Student s);

private :
int marks[5];

};

//Alternative 2
int total_marks(Student s)
{
int total = 0;
for(int i=0;i<5;i++)
total = total + s.marks[i];
return total;
}

int main()
{
        Student* stdprt[550];
}
```

Q2. Ackermann function A(m, n) defined for all m>=0 and n >= 0 as follows

A(m, n) = n+1;	when m = 0
A(m, n) = A(m-1, 1);	when m > 0 and n = 0
A(m, n) = A(m-1, A(m, n-1));	when m > 0 and n > 0

Complete the definition of the function A() given below using recursion and consistent with the main() given that calls it. **(3marks)**

ANSWER :

```
int A( int x, int y)
{

if (x == 0)
return y + 1;

if(x > 0 && y == 0)
return  A(x-1,1);

return A(x-1,A(x,y-1));
















}

int main()
{
  int m, n;
  cin >> m >> n;
  if ( m >= 0 && n >= 0)
       cout << " A[ " << m << ", " << n << " ] : " << A (m, n) << endl;
}
```

ROLL NO : ________________

Q3. Consider the linked implementation of a list of integers given below.

```
class list {
        class node { public:
                  int val;
                  node * next;
             };
         public : node * head;
                list() { head = 0;} // constructor

                bool isempty () { if ( head == 0 ) return true;  else return false; }

                int length ()  { node * p = head;
                         int count = 0;
                         while ( p != 0 ) { count ++; p = p->next; };
             return count;
            }

            list operator+ ( int elm ) // inserts elm at the head of the list
            { node * p = new node;
            if ( p == 0 ) cerr << " fatal error : no space on heap " << endl;
               p->val = elm;
               p->next = head; // connects new node to list
               head = p;
               return *this;
            }

            list  operator+ ( list l)  { skipped  }

             list operator&(int elm )
             { appends elm at the end of list – details skipped  }

            void operator= ( node* p)  { head = p; return ;}

           friend ostream & operator<< (ostream &x, list l)
            { node * p = l.head;
              for ( int i = 1 ; i <= l.length(); i++ )
                   { x << " element no " << i << " is = " << p->val << endl; p = p->next; };
              return x;
            }

    };  // end of class definition
```

(a) The implementation of **operator<<(x, m)** as given in the definition of list above does not display anything if the list argument m is empty. Make minimal changes to the body of this function so that it now displays the following message in the appropriate stream when m is empty, “ List is empty – nothing to display “. When the list m is non-empty it maintains the earlier behavior of this function.

(2 marks)

ANSWER :

```
friend ostream & operator<< (ostream &x, list l)
          { node * p = l.head;
               if(l.length() == 0)
            x << "List is empty – nothing to display";
            for ( int i = 1 ; i <= l.length(); i++ )
               { x << " element no " << i << " is = " << p->val << endl; p = p->next; };
            return x;
          }
```

ROLL NO : ________________

(b) Write a member function overloading unary postfix '++', whose semantics is to add 2 to each element of the list, i.e., (p->data)=(p->data) + 2, where p is a pointer to any element of the list. **(4 marks)**

ANSWER :

```
void operator++(int dummy)
{
   node * p = head;
   for ( int i = 1 ; i <= length(); i++ )
   {
       p->val = p->val + 2;
       p = p->next;
   }
}
```

(c) We need to add a member function to remove an element from a list. This is to be implemented by overloading operator symbol "−" in the form: **list operator− (int n) { …. }.** The semantics of remove is defined as follows. See the table for illustration, L is name of the list.
i) If the element *n* is present in the list, then the first occurrence of *n* is removed from the list,
ii) If *n* is not present in list, then list remains unchanged and a message, " *n* not found, can't remove " is displayed,
iii) If the list is empty, then list remains unchanged and a message " list empty, remove not possible " gets displayed by the function.

Make sure that your code handles all the cases of removal properly, independent of the location of the element to be removed (head, in the middle or at the tail), whether the list is empty or not, whether the element is present in the list or not.

Initial List	Operation	List after operation	Message
L: 5->2->3->2	L − 6	L: 5->2->3->2	6 not found, can't remove
L: 5->2->3->2	L − 5	L: 2->3->2	
L: 5->2->3->2	L − 2	L: 5->3->2	
L: 5	L − 5	L: null (empty list)	
L:null (empty list)	L − 5	L: null (empty list)	list empty, remove not possible

(6marks)

ANSWER :

```
void operator-(int n) // inserts elm at the head of the list
 {
    node* p = head;
    node* prev;

    if(p == NULL) // check if list is empty
    {
      cout << "list empty, remove not possible" << endl;
      return;
    }

     for ( int i = 1 ; i <= length(); i++ )
    {
         if (p->val == n)
         {
           break;
          }
         prev = p;
         p = p->next;
    }

     if(p == NULL) // check if element is not found
           cout << n << " not found, can't remove" << endl;
     else
```

```
        if(p == head) //  if element found is head of list
         {
             head = head->next;
          }
         else   // if element found is other that head
          {
             prev->next = p->next;

           }
  }
```

Question	Total Marks	Marking scheme
Q1 (a) and Q1 (b)	5	1 Mark for correct access specifiers of all elements (Deduct -1\2 mark for each mistake), 1 Mark for correct data types of mobile number and rollno 1 Mark for declaring array of pointers correctly 2 Marks for total_marks() function (friend function OR member function)
Q2	3	1 Mark for each of the three conditions
Q3 (a)	2	1 Mark for checking if list is empty. 1 Mark for printing the message in correct stream. Message should be printed in stream 'x'
Q3 (b)	4	1 Mark for passing dummy argument 1 Mark correct access and manipulation of 'val' attributes 2 Marks for running through the list correctly and correct boundry conditions
Q3 (c)	6	1 Mark each for correct handling of each of five cases illustrated in the question. 1 Mark for absence of segmentation fault (eg accessing fields of a null pointer)