Department of Computer Science and Engineering
CS 101 : Computer Programming and Utilization
SEMESTER END EXAMINATION, AUTUMN 2013– 2014 SESSION

Day, Date & Venue: Saturday, 23/11/2013, HALL 1 to 4
Marks : 80
Duration : 14:00 – 17:00 hrs
Weightage : 40%

QUESTION PAPER AND ANSWER BOOKLET : INSTRUCTIONS

==========

1. PLACE ALL ELECTRONIC GADGETS, IN SWITCHED OFF STATE, IN 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. Assume that the program fragments given in the question paper are all syntactically and semantically correct unless stated otherwise. 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 31 pages and 7 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 : _____

MARKS	Q1	Q2	Q3	Q4	Q5	Q6	Q7	TOTAL
Allotted	**12** (2+4+4+2)	**12** (4+4+4)	**12** (6+6)	**12** (4+4+4)	**12** (6 + 6)	**10** (5 + 5)	**10** (5 + 5)	**80**
Obtained								
Initials of Examiner / Scrutinizer								

Q1. There are four parts of this question. Give brief answers to each.

(a) Given that the integer i has value 2, what is the value of the following expression : **(2 marks)**

i ++ + ++ i;

ANSWER Q1(a):

The value of the expression is 6

Explanation :

pre-increment(++i) is evaluated before the value of expression is computed. Thus value of i becomes 3.

Q1(b) Assume that the following program has been compiled into the executable file, "a.out".

```
int main()
{ int count = 0, i ;
  cerr << " supply the inputs ";
  while ( cin >> i )
 { cout  << " value no. " << ++count << " is = ";
    cout << i << " ";
    if ( count % 3 == 0 )  cout << endl;
 };
 cout << endl;
}
```

Also assume that there is a file named as "**input**", which has the following contents :

10 20 30 40 50

What is displayed on the screen when the following shell command is executed ? Justify your answer. **(4 marks)**

$ a.out < input 2> err | wc -l

ANSWER Q1(b):

Q1(c) Write the output of the following program, given that the l-values (addresses) of i and p are **0xbffff754** and **0xbffff74c** respectively. Justify your answer. **(4 marks)**

```
int main()
{ int i = 10;
  int &r1 = i;
  int *p = &r1;
  int* &r2 = p;
  cout << " i = " << i << " &i = " << &i << endl;
  cout << " r1 = " << r1 << " & r1 = " << &r1 << endl;
  cout <<  " *p = " << *p  << " &p = " << &p << endl;
}
```

ANSWER Q1(c) :

Q1(d) The following program is supposed to find out, maxint, the largest int value on a machine. Without adding any extra variables or lines, fill in the blank spaces so that the function performs its desired task.
(2 marks)

ANSWER Q1(d) :

```
int maxint( )

{    int i = 0;

     int nexti  =  i+1;

     while ( i<nexti )  //  incomplete

     { i = i + 1;

       nexti = nexti+1  //  incomplete

     }

     return  i  // incomplete

}
```

Q2 (a). In the following, incomplete code of a function named as i2s() is given. This function takes an integer as its argument and converts it into a string equivalent. For example the string equivalents of integers 10, -999 and 0 are "+10", "-999" and "0" respectively. Fill in the blank spaces so that the function does its job properly. Do not change the code given. **(4 marks)**

ANSWER 2(a) :

```
string i2s( int i)
{ string s = "";
     string digs[] = {"0","1","2","3","4","5","6","7","8","9"};

     int tmp = i, rem = 0;
```

```
if ( i == 0 ) return "0";  // incomplete

if ( i < 0 )  tmp = i*(-1) OR tmp*(-1)  ;  //  incomplete

  while ( tmp!=0)         // incomplete

  {   rem = tmp % 10 ;

     s =  digs[rem] + s;  // incomplete

     tmp =  tmp / 10 ;
  };

  if ( i < 0)    s = "-" + s ;              //  incomplete

  else
              s = "+" + s;
 return s;
}
```

Q 2(b). Two input files, f1 and f2 are supplied with the following contents :

Contents of file f1 (5 characters): P Q R S T

Contents of file f2 (5 integers) : 1 2 3 4 5

The program given below is compiled into "a.out" and the following command is executed.

```
$ a.out  f1  f2

int main(int argc, char* argv[ ])
{  string s1, s2;
   ifstream in1 (argv[1], ios::in);
   ofstream out1 (argv[1], ios::app);
   ifstream in2 (argv[2], ios::in);
   ofstream out2 (argv[2], ios::app);
   while ( in1 >> s1 ) out2 << s1 << " "; out2 << endl;
   while ( in2 >> s1 ) out1 << s1 << " "; out1  << endl;
```

```
    return 0;
}
```

What are the contents of files f1 and f2 after the execution ? **(4 marks)**

ANSWER 2(b) : Write the contents of f1 and f2, line-wise and give brief reasons.

File f1 : // 2 marks

P Q R S T

1 2 3 4 5 P Q R S T

File f2 : // 2 marks

1 2 3 4 5

P Q R S T

Reason:

The first while loops appends contents of file 1 to file 2. Second while loops updated file 2 and appends it to file1.

Q 2(c) Write the output of the following program and justify your answer. **(4marks)**

```
string s[5] = {"CS101", "CS201", "CS301", "CS401", "CS601"};
string & replace ( int i ) { return s[i]; }
int main() {
for ( int i = 0; i < 5; i++) cout << s[i] << "  ";   cout << endl;
for ( int i = 0; i < 5; i++)    if ( i% 2 == 1) replace(i) = s[5-i];
for (int i = 0; i < 5; i++) cout << s[i] << "  ";   cout << endl;
return 0;
}
```

ANSWER Q2 (c):

CS101 CS201 CS301 CS401 CS601 // 1 mark

CS101 CS601 CS301 CS301 CS601 // 2 marks

Explanation: // 1 mark

The replace function is invoked such that contents of first position are written with contents of fourth position and contents of second position are written with contents of second position.

Q3 (a) Consider the text file, birthdates, given below. Each line of this file has 4 fields, first is an email address, next 3 fields are month, date and year which denote the birthday information of the email addressee. The fields are separated by a single space between them.

abc@gmail.com Feb 12 1995

xyz@yahoo.com Nov 25 1991

axcd@iitb.ac.in Feb 10 1990

rpc@yahoo.com Nov 12 1988

rst@gmail.com Dec 05 1987

psr@ittd.ac.in Dec 31 1992

efg@iitk.ac.in Nov 29 1989

ada@iitb.ac.in Mar 01 1983

Consider the shell program, saved in file, bday.sh, and given below. Assume that the output of date command is Fri Nov 23 14:00:00 IST 2013 for this problem. You have to write information as asked at the three marked points. **(2 + 2 + 2 = 6 marks)**

```
# shell program for processing birthdates file and generating messages ;
# file bday.sh
date=`date`     #   Take date as Sat Nov 23 14:00:00 IST 2013
echo $date > file1
read day mon dat time zone year < file1
cat birthdates  |
     grep " $mon " |
       sort -nk3 > file2
# marked point 1
cat file2 | cut -d' ' -f3 > f1
cat file2 | cut -d' ' -f2 > f2
cat file2 | cut -d' ' -f1 > f3
paste -d' ' f1 f2 f3
# marked point 2
years=0
while read email mn dt yr
do
 if test $dt -ge $dat
 then
 years=`expr $year - $yr`
 echo " Send  mail to  $email  on $mn  $dt "
 echo "     Happy Birthday on attaining the age of $years Years"
 fi
done < file2
# marked point 3
rm file1 file2
exit 0
```

ANSWER Q3(a) : **Show the contents of file2 at the marked point 1**

rpc@yahoo.com Nov 12 1988

xyz@yahoo.com Nov 25 1991

efg@iitk.ac.in Nov 29 1989

Show the display on screen at the marked point 2

12 Nov rpc@yahoo.com

25 Nov xyz@yahoo.com

29 Nov efg@iitk.ac.in

Show the additional display of file2 at the marked point 3 (do not show the display shown at marked point 2 again)

Send mail to xyz@yahoo.com on Nov 25

Happy Birthday on attaining the age of 22 Years

Send mail to efg@iitk.ac.in on Nov 29

Happy Birthday on attaining the age of 24 Years

OUTPUT : marked point 1: (2 marks, deduce marks based on the number of errors)

```
# file bday.sh
date=`date`                          #   Take date as Sat Nov 23 14:00:00 IST 2013
echo $date > file1                   #    file1 contents :  Sat Nov 23 14:00:00 IST 2013
read day mon dat time zone year < file1   # assigns values to user defined shell variables, day=Sat, mon=Nov, dat=23,
                                     # concatenates file birthdates to screen which is piped to grep filter
    grep " $mon " |                  # grep pulls out those lines that match $mon that is Nov anywhere in the line and pipes to sort
        sort -nk3 > file2                    # lines 2, 4 and 7 are sorted by sort using numeric sorting in ascending order on column 3 and saves in file2
```

Lines 2, 4 and 7 after sorting on column 3 becomes lines 4, 2 and 7 which is shown below :

rpc@yahoo.com Nov 12 1988

xyz@yahoo.com Nov 25 1991

efg@iitk.ac.in Nov 29 1989

Marked point 2 : The shell commands that are involved are given below :

```
cat file2 | cut -d' ' -f3 > f1        # cuts col3 from file2, the date field and stores it in file f1; f1 has 3 lines : 12\n25\n29\n  (where \n is newline)
cat file2 | cut -d' ' -f2 > f2        # cuts col2 from file2, the month field and stores it in file f2; f2 has 3 lines :
```

Nov\nNov\nNov\n

cat file2 | cut -d' ' -f1 > f3 # cuts col1 from file2, the email address field and stores it in file f3; f2 has 3 lines : similar to above

paste -d' ' f1 f2 f3 # pastes the columns from files f1, f2 and f3 in that order and places a single space (-d' ') between the values in a line

The output at marked point 2 is given below : **(2 marks, deduce marks based on the number of errors)**

12 Nov rpc@yahoo.com

25 Nov xyz@yahoo.com

29 Nov efg@iitk.ac.in

Marked point 3 : the shell commands that generate the output at marked point 3 are

years=0 # define a var years and set it to 0

while read email mn dt yr # read reads a line of input from file2 and assigns the 4 strings respectively to four shell variables

do # email, mn, dt and yr that store the address, month, date and year as strings

if test $dt -ge $dat # the value of $dt is compared with today's date $dat

then # this part of code is executed only if the date in file is greater than today's date (all except 12 passes)

years=`expr $year - $yr` # find the difference between the birth year, $year and $yr

echo " Send mail to $email on $mn $dt " # message to be displayed showing email, month and date

echo " Happy Birthday on attaining the age of $years Years" # message that displays the age on the message

fi

done < file2

The output displayed therefore because of the lines above are : **(2 marks, deduce marks based on the number of errors)**

Send mail to xyz@yahoo.com on Nov 25

Happy Birthday on attaining the age of 22 Years

Send mail to efg@iitk.ac.in on Nov 29

Happy Birthday on attaining the age of 24 Years

Q3(b) You have to write a shell program, dirstats.sh, that gives information about directories as described here. A partial listing given by, **ls -l** /, for the contents of root directory (/) is given for illustration. Note the second field of the listing, called **links**. A directory entry with links equal to 2 denotes an empty directory. The number of entries (an entry is a file or a directory) in a directory is given by the expression **links – 2**; thus the two directories **bin** and **cdrom** are empty while **dev** and **etc** directories have 14 and 135 entries respectively.

drwxr-xr-x **2** root root 4096 Oct 4 05:19 bin

drwxr-xr-x **2** root root 4096 Oct 1 22:22 cdrom

drwxr-xr-x **16** root root 4380 Nov 22 15:07 dev

drwxr-xr-x **137** root root 12288 Nov 22 15:07 etc

The usage of the shell program to be written is as follows : $. / dirstats dir_name 5

The first argument is a directory name, specified either by absolute path or relative path from current directory ($PWD) and the second argument is a strictly positive integer. You may assume that both the command line arguments, dir_name and number are correct and need not check for their validity. The program should perform two tasks, (a) List all the empty directories in the directory dir_name, and (b) List the top number 5 (number specified as second argument) directories in decreasing order of their number of entries (that is links value). Sample expected output for a certain directory, say xyz, when invoked with **$ ./dirstats xyz 4** is shown below. The empty directories may be listed in the same order as that given by **ls -l** .

List of empty Directories :

drwxr-xr-x 2 root root 4096 Oct 4 23:23 xml

drwxr-xr-x 2 root root 4096 Oct 4 23:04 ssh

drwxr-xr-x 2 root root 4096 Oct 4 23:04 init

drwxr-xr-x 2 root root 4096 Oct 4 23:04 default

List of top 4 populated Directories

drwxr-xr-x 12 root root 4096 Oct 4 23:23 texmf

drwxr-xr-x 10 root root 4096 Apr 24 2013 X11

drwxr-xr-x 8 root root 4096 Apr 24 2013 ppp

drwxr-xr-x 8 root root 4096 Apr 24 2013 apparmor.d

ANSWER Q3(b) : **(6 marks)**

```
#  usage ./dirstats dir_name   count
#  usage dirstats dir  count
#  Given dir - a directory name and count - a number
#  a) list the directories that are empty (link value is 2), one per lien
#  b) count number of directories that have largest no of entries, one per line
#     in descending order of link value
dir=$1                          # saving $1 the first argument – not mandatory, may use $1 directly
ls -l  $dir  |                  # displays the output of ls in long form is piped to grep
   grep '^d' |                  # grep pulls out all lines starting with d and pipes to sort
     sort -r n  -k2  > file     # sort sorts line-wise reverse numeric order (-r n) on column 2 (-k2) which is the
number of links and saves it in file
```

 # Partial marking step 1 : anything equivalent to above, use of a temporary file is not mandated : pull out the directory lines, sort the lines

 # in decreasing links field value : 2 marks

```
echo " List of empty Directories : "
 while read perm links rest   # while loop is  on read and read is reading from file2 (< file)
 do                           # first field is saved in var perm, 2nd field  field in var links and  the rest of string in te line in var rest
  if test $links -le 2        # if var links is <= 2 ( check for empty directory )
  then
    echo $perm $links $rest   # display the empty directories
  fi
done < file
```

 # Partial marking step 2 : Processing the sorted lines to identify lines that have link value <=2 (may use == in place of <=)

 # and correctly identifies such lines and displays : 3 marks

```
echo " List of top $2 populated Directories"  # shell replaces $2 with the integer value given in command line arg 2
cat file | head -n$2 # file is already sorted in decreasing value of link field head pulls out $2 number of lines from the start of the file (-n$2)
```

 # Marking part 3 : 1 mark

Caution : Alternate solutions are possible – please give appropriate part marks depending on how much is done correctly.

Q4. Examine the following program.

```
class A
{ public: virtual int func(int i) { cout<<"In base class A func " ; return ++i;};
          void f1 (int i) { cout << func(i)<< endl;};
};
class B : public A
{ public: virtual int func(int i) { cout<<"In class B func "; return i*i;};
          void f1 ( int i) { cout << func(i) << endl;};
};
class C: public B
{ public:  int func(int i) { cout<<"In class C func "; return i*11;}
```

```
};
class D : public C
{ public: int func(int i)  { cout<<"In class D func "; return i*12;}
};
class E : public A
{ public: int func(int i)  { cout<<"In class E func "; return 3*i;}
};
class F : public E
{ public: int func(int i)  { cout<<"In class F func "; return 4*i;}
};
int main()
{  A *a[5], aa;
   B *b[3], bb;
   C cc;  D dd; E ee; F ff;
  a[0] = &aa; a[1] = &bb; a[2] = &dd; a[3] = &ee; a[4] = &ff;
  b[0] = &bb; b[1] = &cc; b[2] = &dd;
  for ( int i = 0 ; i < 5 ; i++ ) a[i]->f1(i);
  for ( int i = 0 ; i < 3 ; i++ ) b[i]->f1(i);
}
```

Q4(a) Write the output of the program and for each call to f1(), identify the function definition the call it is resolved to. **(4marks)**

ANSWER Q4(a) :

In base class A func 1

In class B func 1

In class D func 24

In class E func 9

In class F func 16

In class B func 0

In class C func 11

In class D func 24

Q4(b) If the virtual keyword associated with the func() in class A is dropped, the new definition of class A is as shown below, and no other change in the rest of the program, write the output of the changed program justifying your answer. **(4 marks)**

class A

```
{ public:   int func(int i) { cout<<"In base class A func " ; return ++i;};
            void f1 (int i) { cout << func(i)<< endl;};
};
```

ANSWER Q4(b) :

In base class A func 1

In base class A func 2

In base class A func 3

In base class A func 4

In base class A func 5

In class B func 0

In class C func 11

In class D func 24

Q4(c) Examine the following program given below. You have to write the output in the correct sequence of constructors and destructors called. Justify your answer in brief. **(4marks)**

```
class A { int i;
  public:
        A(int ii) : i(ii) {cout << " constructor of A " << endl;}
       ~A() {cout << " destructor of A " << endl;}
};
class B : public A
{   A a;
    int i;
   A aa;
 public:
   B(int ii) : a(ii) , aa(ii),i(ii),A(ii) { cout << " constructor of B" << endl;}
  ~B() { cout << " destructor of B" << endl;}
};

class C : public A
{  A a;
   int i;
  A aa;
  public:
    C(int ii) : i(ii) , aa(ii),A(ii),a(ii) { cout << " constructor of C" << endl;}
   ~C() { cout << " destructor of C" << endl;}
};
int main()
{   B b(4);
    C c(5);
}
```

ANSWER Q4(c) :

constructor of A
constructor of A
constructor of A
constructor of B
constructor of A
constructor of A
constructor of A
constructor of C
destructor of C
destructor of A
destructor of A
destructor of A
destructor of B
destructor of A
destructor of A
destructor of A

Q5(a). Given all the real roots of a polynomial, it is required to construct the polynomial. For example, given 3 roots, r_0, r_1 and r_2 , the constructed polynomial p(x) would be of degree 3 and have 4 coefficients, a_0, a_1 ,a_2, and a_3; where $p(x) = a_0 + a_1x + a_2x^2 + a_3x^3$, and $a_3 = 1$.

The relation between a coefficient and the roots are also well known, such as $a_2 = -(r_0 + r_1 + r_2)$; $a_1 = r_0 * r_1 + r_0 * r_2 + r_1 * r_2$; $a_0 = -r_0 * r_1 * r_2$

You have to complete the body of the function create_poly(float*, int) which uses the principle mentioned above to construct the n+1 coefficients of a polynomial of degree n, whose n roots are given. Do not introduce any new variables and do not change any part of given code. **(6 marks)**

ANSWER :

```
// root [ ] is the array of roots, max 100 roots assumed
// coeff [ ] is coefficient array created by this function on the heap and its pointer is returned back
// n is the number of roots supplied
float * create_poly( float root [ ], int n)

{ float *coeff = new float[101]; // assume n <= 100

  for ( int i = 0; i < n; i++ ) coeff[i] = 0.0;

                                // write your code here

 for ( int i = 0; i < n; i++ )

     for ( int j =          ;          ;  j++)      //
incomplete

    // write your code here



 // now fix the signs

   int sign = -1;

   for ( int i = n; i >= 0; i - -)


   {

   };

   return coeff;

}
```

```
float * create_poly( float *root, int n)

{ float *coeff = new float[101]; // assume n <= 100

  for ( int i = 0; i < n; i++ ) coeff[i] = 0.0;

   coeff[n] = 1.0;                          …..... 0.5 m

   for ( int i = 0; i < n; i++ )

     for ( int j = n-i-1 ; j < n ; j++)    ..... 0.5 mark


          coeff[j] = coeff[j] + root[i] * coeff[j+1];   .....
4 marks


 // now fix the signs

   int sign = -1;

   for ( int i = n; i >= 0; i--)      .

  { sign = - sign;   …..... 0.5 m

    coeff[i] = sign * coeff[i];    ..... 0.5 m
```

```
};
return coeff;
}
```

Explanation :

1. No of coefficients when the equation has n roots, {r_0, r_1, r_2,r_3 , ... , r_{n-1}} are n + 1. We shall use the notation given below for expressing a n-th degree polynomial

$a_nx^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + ... + a_2x^2 + a_1x + a_0$;

2. The coefficient, a_n, of the highest exponent (x^n) is always 1 ; the constant term, coefficient a_0, is always the product of all the roots, i.e., $r_0 * r_1 * r_2 * r_3 * r_4 * ..... * r_{n-1}$

Incomplete ….....

Q5(b) Consider the linked list implementation of a queue data structure. It is required that the class Queue be extended to a template class such that the queue can be used to hold scalar values, other than int, also. **(6 marks)**

```
class Queue{
   class node { public:
                int val;
                node * next;
    };
   public : node * front, *rear;
         Queue() { front = 0; rear = 0;} // constructor
        bool isempty () {
        if ( front == 0 && rear == 0) return true;   else return false; }

         bool insert (int elm )
         { node * p = new node;
           if ( p == 0 )
           {cerr << " fatal error : no space on heap " << endl; return
false; }
           p->val = elm;   p->next = 0; // rear of the changed queue
           if (rear != 0) rear->next = p;
           rear = p;
           if (front == 0 ) front = p;
           return true;
         }

        int remove ()
         { if ( isempty() == true )  return -1; // queue is empty
           int res = front->val;  // value at front of queue
           front = front->next; // change front
           if (front == 0) rear =0;
           return res;
         }
         void display()
         { cout << " Elements in queue from front to rear : " ;
           if ( isempty() == true)  cout << " empty : nothing to display
\n";
            else
            { node *p = front;
              while (p != 0)  {cout << p->val << " "; p =p->next;}
              cout << endl;
            }
         }
};
```

```
template <class T>                    //  change 1
 class Queue{
      class node { public:
                  T val;                  //  change 2
                  node * next;
      };
   public : node * front, *rear;
              Queue() { front = 0; rear = 0;} // constructor
     bool isempty ()
         { if ( front == 0 && rear == 0) return true; else return
false; }
    bool insert (T elm )                    //  change 3
         { node * p = new node;
         if ( p == 0 )
           {cerr << " fatal error : no space on heap "<< endl; return
false; }
           p->val = elm;
           p->next = 0; // rear of the changed queue
           if (rear != 0) rear->next = p;
           rear = p;
           if (front == 0 ) front = p;
           return true;
         }
      T remove ()                           //  change 4
         { if(isempty() == true) return res; // queue is empty
           T res = front->val;  // value at front of queue   //  change
5
           front = front->next; // change front
           if (front == 0) rear =0;
           return res;
         }

        void display()
        {  cout << " Elements in queue from front to rear : " ;
           if(isempty() == true) // queue is empty
              cout << " empty : nothing to display \n";
           else
            { node *p = front;
              while (p != 0)  {cout << p->val << " "; p =p->next;}
              cout << endl;
            }
         }
};
```

Identify the parts of the above definition that must be changed for the required conversion to a template definition with reasons. Write the changed part of the definition only.

ANSWER Q5(b).

The new design is given in column 2 above. The changes that are required are marked in bold.

All 5 changes are done : full marks ; extra changes that do not effect the correctness of the template design are ok.

Change 5 may be given more weightage, may be 2 marks while the others are 1 mark each (suggestion)

The questions Q6 and Q7 refer to an academic software system (a simplified version of ASC), described here. The input to the system are in the form of a text file. The software is targeted for all 4/5 year program UG students at the end of first two semesters of their program.

Specifications of IITB First year of 4/5 yr Undergraduate Program :

1. Total stipulated credits at the end of first year : **67.**

2. A **course** is specified by a **triple** comprising i) course code, ii) course credits, and iii) grade obtained. Semester wise course list of a student is a list of all courses credited in a given semester, where each course is represented by a triple.

2. A **Cleared course** is a course for which a student has acquired a pass grade (all grades except FR are pass grades, **FR is the only fail grade**).

GRADE	AP	AA	AB	BB	BC	CC	CD	DD	FR
POINTS	10	10	9	8	7	6	5	4	0

3. A Course in which a student has got FR grade is called a **Backlog course**. A backlog course for all calculation purposes is counted exactly once independent of the number of times the student gets FR grade in this course. A backlog course is converted to a cleared course once the student passes the course.

4. Outstanding credits is the difference between the stipulated credits 67 and the sum of credits of Cleared courses and Backlog courses.

5. A Cleared course can not be registered again. However, there is no limit on the number of times a backlog course may be registered for.

6. At the end of each semester, a Semester Performance Index (SPI) of a student for that semester is calculated as follows :

Let the C_i, R_i and G_i denote respectively the course code, number of credits and grade acquired, for the i^{th} course in the semester, $1 \le i \le n$; then the semester credit points earned = $\sum_{i=1}^{i=n} R_i * G_i$,

the semester credits = $\sum_{i=1}^{i=n} R_i$ and $SPI = \frac{semester\ credit\ points}{semester\ credits}$

7. At the end of each semester, a Cumulative Performance Index (CPI) of a student is calculated as follows. Let the C_k, R_k and G_k denote respectively the course code, number of credits and grade acquired, for all the courses in all the semesters starting from 1 to the semester under concern; where a) each course is listed exactly once and b) for a course that is listed more than once, the data of the most recent semester is retained and the others are ignored.

Cumulative credit points = $\sum_{k=1}^{k=m} R_k * G_k$,

the cumulative credits = $\sum_{k=1}^{k=m} R_k$ and $CPI = \frac{cumulative\ credit\ points}{cumulative\ credits}$

Example : Consider the semester course list for a student as given below.

	COURSES IN SEMESTER 1							COURSES IN SEMESTER 2					
Code	AA001	AB001	AC001	AD001	AE001	AF001		BB002	AC001	BC002	AF001	BD001	BE002
Credits	6	8	6	4	3	6		6	6	5	6	3	8
Grade	AB	CD	FR	BB	BC	FR		AB	DD	BB	FR	AA	FR

The calculations of SPI, CPI, credit sum of cleared courses, backlog information and outstanding credits are shown below.

	Semester 1	Semester 2
Credits	6 + 8 + 6 + 4 + 3 + 6 = 33	6 + 6 + 5 + 6 +3 + 8 = 34
Credit Points	54 + 40 + 0 + 32 + 21 + 0 =147	54 + 24 + 40 + 0 + 30 + 0 = 148
SPI	147 ÷ 33 = 4.454	148 ÷ 34 = 4.352
CPI	147/33 = 4.454	(147+148) ÷ 55 = 5.363
Credit sum of Cleared courses	21	20
Backlog Courses information :	2 courses; AC001 6 FR AF001 6 FR	2 courses; AF001 6 FR BE002 8 FR
Outstanding Credits	Not Applicable	12 credits (67 -(41 + 14))

Q6. A part of the design of the academic system to handle first year course and related information is given below.

```
class coursedata
{ public :
  string code;          // course code
```

```
  short credits;          //  course  credits
  string grade;           //  course grade
  coursedata operator=(coursedata y)  { code = y.code; credits = y.credits; grade = y.grade; return *this; }
  friend ifstream& operator>> (ifstream& , coursedata &);
};


ifstream& operator>> (ifstream& x, coursedata &cdt)

 { x >> cdt.code >> cdt.credits >> cdt.grade;  return x;  }


class student_record

{  public :
    string roll;
    coursedata sem1[10];          //  course data for sem1; sem1[0].credits gives the actual number of
                                  //  courses registered; sem1[1] onwards are the actual course data
    coursedata sem2[10];          //  similar for sem2 courses
    coursedata merged[20];        // set up by mergesemdata()
    coursedata backlogs[10];      // populated by backlogcal()
    float spi[3];                 // calculated by spical()
    float cpi[3];                 // calculated by cpical()
    student_record(){   };
    void mergesemdata();
    void spical();
    void cpical();
    void backlogcal();
    friend ifstream& operator>> (ifstream& x, student_record &std);
  };
```

You have to write the body of a few functions listed below.

Q6(a) Complete the body of he following function whose task is to return the integer value corresponding to the grade supplied through its argument. **(4 marks)**

ANSWER Q.6(a) :

```
short grade2points(string gr)
{ short points = 0;
   if (gr == "AP") return 10;
  else if (gr == "AA") return 10;
  else if (gr == "AB") return 9;
  else if (gr == "BB") return 8;
  else if (gr == "BC") return 7;
  else if (gr == "CC") return 6;
  else if (gr == "CD") return 5;
  else if (gr == "DD") return 4;
  else if (gr == "FR") return 0;
  return points;
}
```

Marking scheme : Other equivalent forms are acceptable. ½ mark for handling of each correct grade. There are 9 grades, so all correct gets full credit. ½ mark deducted for not returning the value for every single grade.

(b) The data for students course records are supplied through an input file, **acadrecords.txt**, sample contents are given below:

```
900
140001001 6 6
AA001 6 AB   AB001 8 CD   AC001 6 FR  AD001 4 BB   AE001 3 BC   AF001 6 FR
BB002 6 AB   AC001 6 DD   BC002 5 BB   AF001 6 FR  BD001 3 AA   BE002 8 FR
```

The first line gives the number of students whose records are supplied. The second line gives the roll number and the number of courses undertaken in sem1 and sem2 respectively. The subsequent two lines give the data about 6 courses in sem1 and sem2 respectively, each semester data on a line. The data of a course is its code, credits and grade as explained earlier. The pattern shown in the kines 2 to 4 are repeated for every student in the institute. You are required to complete the body of the overloaded operator>> for the student_record class. Its use could be in the following context.

```
//  intended use of operator>> for student_data
ifstream in ("acadrecords.txt", ios::in);
int students; // number of students in input file (first data)
in >> students;
// read from acadrecords.txt ; read number of students;
student_record firstyear[SIZE];
for (int i = 1; i <= students; i++)
{ // local info for each student
   in >> firstyear[i];
  //  rest of processing
```

The operator>> is supposed to read the first line of a student record, save the first value in data member **roll,** the next two numbers denting the number of courses done in sem1 and sem2,in the data members **sem1[0].credits** and **sem2[0].credits** respectively. The actual course data are stored in the 3 fields of sem1[1] to sem1[sem1[0].credits] and similarly for sem2. Essentially the triples sem1[0] and sem2[0] are dummy objects and are not used to store actual course triples, it also keeps indexing easy, as the n courses are accessed through sem1[1] .. sem1[n]. **(6 marks)**

```
ifstream& operator>> (ifstream& x, student_record &std)
 { // processing  information for each student
  x >> std.roll >> std.sem1[0].credits >> std.sem2[0].credits;                  // 2 marks
  for (int k = 1; k <= std.sem1[0].credits; k++)    { x >> std.sem1[k]; }       // 2 marks
  for (int k = 1; k <= std.sem2[0].credits; k++)    { x >> std.sem2[k]; }       // 2 marks
```

return x;

}

Explanation : The implementation of this operator is to be examined in the given context, which is specified through the following statements provided in the question:

in >> students; // reads the first line of output leads to students = 900

for (int i = 1; i <= students; i++) // Loop iterates over students => body does the processing for one student

{ in >> firstyear[i]; // Inference : this statement has to read the entire information about a student which are the contents of lines 3 to 5

The 3 values on line 2 are roll number, number of courses in sem1 and sem1 respectively of a student; the corresponding data members have to be initialized after read. The **operator>>** with ifstream object **in** is the correct combination (getline or other forms may be correct but not warranted). Hence the natural choice is the following. The use of the field sem1[0].credits for string the number of courses listed in sem1 (and similarly for sem2) is mentioned in the question. Note that the access of data members involved in this statement passes through because of public specification.

x >> std.roll >> std.sem1[0].credits >> std.sem2[0].credits; **// 2 marks**

Line 2 gives as many course triplets as std.sem1[0]. credits and definition of an operator>> is given for the class coursedata, because of which the following statement

for (int k = 1; k <= std.sem1[0].credits; k++) { x >> std.sem1[k]; } **// 2marks**

reads the course triplets for each course in sem1. Again this is convenient and a desirable manner for reuse of existing design. The entire line given in the input below is read and the data members of sem1 of std[1] properly initialized by this for-loop.

AA001 6 AB AB001 8 CD AC001 6 FR AD001 4 BB AE001 3 BC AF001 6 FR

The second for-loop given in the answer similarly reads the data for sem2 and does the needful.

Marking Scheme : As shown above, 2 marks each for the subparts

Alternate : Since all members are public **x >> std.sem1[k];** may be replaced by **x >> std.sem1[k].code >> std.sem1[k].credits >> std.sem1[k].grade;** and the code will still work. Essentially the use of overloaded >> for coursedata objects can not be mandated. Similarly for reading sem2[] courses. Do not reduce marks for this equivalent form. Deduce ½ mark for each other error.

Q7 (a). You have to complete the two functions that are described below. Note that each part is independent of the other.

bool check (coursedata x[], int size, coursedata y)

{ bool val = false;

```
  for (int i = 1; i < size; i++)
  { if (y.code == x[i].code)  { x[i] = y; return true; }
  };
  return val;
}
void student_record::mergesemdata()
{ int courses = sem1[0].credits;
  for (int i = 1; i <= courses; i++) { merged[i] = sem1[i]; }
  int count = courses;
  for (int k = 1 ; k <= sem2[0].credits ;  k++)
  { if (check(merged, count, sem2[k]) == false)  { count++;  merged[count] = sem2[k]; }  };
  merged[0].credits = count;
  return;
}
```

The member function mergesemdata() examines the two semester course data in sem1[] and sem2[] and merges the data so that each course has a unique entry. In case of multiple entries of the same course, the latest entry is retained and the earlier contents are overwritten.

(a) You have to write the member function cpical() whose task is to calculate cpi at the end of sem1 and sem2 and save these values in the data members cpi[1] and cpi[2] respectively. Note that cpi[0] is not in this design. You may assume that other data members have been populated while computing cpi. **(5 marks)**

ANSWER Q 7(a) : The member function mergedsemdata()

```
void student_record::cpical()
{ // populate cpi[] array for indices 1 and 2;  cpi[0] = 0.0;
  int courses = merged[0].credits;        //  number of distinct courses after merging sem1 and sem2 data, in function mergedsemdata()
                                          //  1 mark
  short cum_cr_points = 0, cum_credits = 0;
  for ( int k = 1; k <= courses; k++)                  // iterate for each course in merged[] array
  { cum_cr_points += merged[k].credits*grade2points(merged[k].grade);  // calculate credit points earned per course and accumulate
    cum_credits += merged[k].credits;                  //  sum of credits
  }                                                    //  3 marks for correct loop
  cpi[1] = spi[1];                                     //  cpi of first sem is same as spi of first sem – marks 0.5
  cpi[2] = cum_cr_points / float(cum_credits);         //  cpi of second sem          –  marks 0.5
  return;
```

}

1 mark for figuring out that number of distinct courses are available in the data member merged[0].credits

3 marks for accessing the credits and grades earned in various courses, calculating accumulated credit points and total credits registered for.

0.5 marks for cpi[1] for cpi at the end of sem1 (either use cpi[1] = spi[1] or directly compute the sum as in

```
int courses = sem1[0].credits;
 short sem_cr_points = 0, sem_credits = 0;
 for (int i = 0; i < courses; i++)
 { sem_cr_points += sem1[i].credits*grade2points(sem1[i].grade);
   sem_credits += sem1[i].credits;
 }
 spi[1] = sem_cr_points / float(sem_credits);
```

0.5 marks for cpi[2] for cpi at the end of sem2

Q7(b). You have to write the body of the member function backlogcal() of the student_record class. The task of this function is to populate the backlog[] data member. Similar to the other arrays used, backlog[0] does not store a triple for any backlog course; instead backlog[0].credits is used to save the number of backlog courses at the end of sem2. This function assumes that the data member merged[] has already been populated and processes it to populate the data member triples, backlog[1] to backlog[backlog[0].credits] and the values of a triple in are in the same order, code, credits and grade for every backlog course at the end of semester2. **(5 marks)**

```
void student_record::backlogcal()
{ // this member function sets up the array backlogs[]; backlog[0].credits gives number of backlogs;
   short index = 0;
   for (int i = 1; i <= merged[0].credits; i++)
     if (merged[i].grade == "FR")     backlogs[++index] = merged[i];   // triple assignment
   backlogs[0].credits = index;
  return;
}
```

4 marks for the setting up the backlogs[] array for all courses with FR grades and assign all the three fields, code, credits and grade, and

1 mark for saving the number of backlog courses in the data member backlogs[0].credits