Indian Institute of Technology Bombay, Mumbai
Department of Computer Science and Engineering

CS 101: Computer Programming - Endsem: 23 Apr 2014, 17:30 – 20:30
One sheet of handwritten notes permitted. No photocopies. Weightage: 35%, Max Marks – 70

NOTE: Answer in the space provided. Extra pages will not be provided. In case of any doubts, make assumptions, state these with justification and proceed. Clarifications are not possible.

Write your Roll Number here: ____________________ Write your Lab Batch here: _________

Question	1a	1b	1c	2a	2b	3a	3b	4a	4b	Total
Marks										

Q1a: What is the output of the following program? [**3 Marks]**

```
int a[5] = {10,3,5,1,2};

for (int i=4; i>0; i--) {
   a[i] += a[i-1];
   cout << a[i] << " ";
}
```

Space for Rough Work:

Marking Scheme:
3 Marks for fully correct answer
1 Mark for partially correct answer

Output: 3, 6, 8, 13

Q1b: What is the output of the following program? [**3 Marks]**

```
string c = "hi";
while (c.length() < 6) {
c += c;
cout << c << "\n";
}
```

Space for Rough Work:

Marking Scheme:
3 Marks for fully correct answer
1 Mark for partially correct answer

Output: hihi
hihihihi

Q1c: What is the output of the following program? [**4 Marks]**

```
void something (int* p, int* &q) {
  *p = 5;
  p += 3;
  *p += *q;
  cout << *p << " " << *q << "\n";
  q--;
  return;
}


int main() {
  int a[4] = {10,20,30,40};
  int* b = a;
  int* c = &a[2];
    something (b, c);
    cout << *b << " " << *c;
return 0;
}
```

Space for Rough Work:

```
a = 10 20 30 40
        5
                70
main
b = 10 5
c = 30 20

something
p = 10 5 70
<---q is the same as c
```

Output:

```
70 30
5 20
```

Marking Scheme:
4 Marks for fully correct answer
2 Marks for partially correct answer

Q2a: We know that: $e^x = 1 + x + x^2 /2 + x^3 /3! + x^4 /4! + . . .$
Write a program that takes n and x as input, and outputs an estimate of e^x using first n terms of this formula.
[10 Marks]

```
#include <iostream>
using namespace std;

int main() {
float x;
int n;
  cout << "Enter value of x : "; cin >> x;
  cout << "Enter value of number of terms (n) : "; cin >> n;

float term = 1.0;
float cterm = 1.0;

for(int i=1; i < n; i++) {
            cterm = cterm * x/(float)i;  // (float) is optional
            term = term + cterm;
}

cout << "value of e^x = " << term << endl;
return 0;
}
```

Run: endsem-ex.cpp
Note: the most efficient way to compute the series is as above. However, some students may have used the library function pow(x, i) to compute x^i, and a nested for loop to compute i!. That is also fine.

Marking Scheme:
1 Mark – Header declarations.
2 Marks – Input statements.
1 Mark – Output statement.
6 Marks – Computation, as per the code or the note, above.

For any other approach, check logic and give marks accordingly.
If only pseudo-code is given, evaluate the question out of 5 Marks.
Partial marking as per TA discretion.

Write your Roll Number here: ___________________

Q2b: Two words, *w1* and w2*,* are said to be anagrams of each other if word *w2* can be formed by rearranging letters of word *w1*. For example, “pan” and “nap” are anagrams, and so are “listen” and “silent”. Write a function *isanagram()* which checks if *w1* and *w2* are anagrams of each other. You may use the C++ string class or the C strings library functions. **[10 Marks]**
(If you do not know how to do this in either way, you may write pseudo-code, for 50% reduced marks.)

```
#include <string> // C++ string class

int isAnagram(string s1, string s2) {
  int len1, len2;

  len1 = s1.length(); len2 = s2.length();
  if (len1 != len2) return 0; //Not anagram

  for (int i=0;i<len1;i++) {
            char a = s1[i];
            int pos = s2.find(a, i);
            if (pos == -1) return 0;
            else {s2[pos] = s2[i]; s2[i] = a;}
  }
return 1;
}
```

Marking Scheme:
1 Mark – Function arguments passed correct
2 Marks – Exit if string lengths are not equal
2 Marks – Loop setup
2 Marks – Condition check to exit if not anagram
2 Marks – Correct processing inside loop
1 Mark – Readability of code

```
#include <cstring> // C string library

int isAnagram2(char str1[], char str2[]) {
  int len1, len2;

   len1 = strlen(str1); len2 = strlen(str2);
   if (len1 != len2) return 0; //Not anagram

   for (int i=0; i<len1; i++) {
            char a = str1[i];
            int j=i;
            bool found = false;
            while (j<len2) {
                        if (str2[j] == a) {found = true;
                           str2[j] = str2[i]; str2[i] = a; break;}
                        else j++;
            }
            if (found == false) return 0;
   }
return 1;
}
```

Marking Scheme:
1 Mark – Function arguments passed correct
2 Marks – Exit if string lengths are not equal
2 Marks – Loop setup
2 Marks – Condition check to exit if not anagram
2 Marks – Correct processing inside loop
1 Mark – Readability of code

Run: endsem-strings.cpp
Note: Some alternative approaches are:
(a) For each character in the first string, delete its first occurrence in the second string.
(b) Sort the both strings and compare.

For any other approach, check logic and give marks accordingly.
If only pseudo-code is given, evaluate the question out of 5 Marks.
Partial marking as per TA discretion.

Q3a: Write a program to open a text file called "xyz.txt" and output the longest word in the file**.** If the longest word is not unique, i.e., the file has many words that have same 'longest' length, output **all** these words, in the order in which they appear in the file. **[10 Marks]**
You may assume that the file contains only characters and adjacent words are separated by blank-spaces.
You must use the C++ #include<fstream> library.
(If you do not know how to do this in C++, you may write pseudo-code, for 50% reduced marks.)

```
#include <iostream>
#include <fstream>
using namespace std;

int main () {
  fstream file;
  string word;

  int nWords = 0;
  const int n = 10;
  string longest[n]; //Assuming there wont be more than n longest words
  longest[0] = " "; //Initializing the first word in longest, for first comparison

  file.open ("xyz.txt", fstream::in); // Assuming file exists, not performing error check
  while (file >> word) {
            if (word.length() > longest[0].length()) { //set longest[0] to word, reset counter
                      longest[0] = word; nWords = 0;
            }
            else if (word.length() == longest[0].length()) { //increment counter, store word
                      nWords++; longest[nWords] = word;
  }
  }
  cout << "The longest word in the File has " << longest[0].length() << " characters\n";  //This is optional
  cout << "Number of longest words in the File is = " << nWords+1 << endl;               //This is optional
  for (int i=0; i<nWords+1; i++) cout << longest[i] << endl;
  file.close();
}
```

Run: endsem-files.cpp

Marking Scheme:
2 Marks – Headers and variable declarations.
2 Marks – File open and close statements.
1 Mark – Setup of loop to go through file.
1 Mark – Condition check to see if current word read is the longest.
1 Mark – Keeping track of the current longest word.
2 Marks – Use of array or some other mechanism to store multiple longest words.
1 Marks – Output statement(s) – similar to above.

For any other approach, check logic and give marks accordingly.
If only pseudo-code is given, evaluate the question out of 5 Marks.
Partial marking as per TA discretion.

Write your Roll Number here: ____________________

Q3b: Write a function *merge()* that takes two linked lists, each of which is sorted in increasing order, merges them into one list which is in increasing order, and returns this merged list. You should reuse the nodes in the input lists to create the merged list. **[10 Marks]**
(If you do not know how to do this in C++, you may write pseudo-code, for 50% reduced marks.)

```
struct node { // Assuming each node of the linked lists has the following structure
        int num;
        node * next;
};

node* merge(node* L1, node* L2) {//merge the nodes in L1 and L2
        node *head, *tail;

        if (L1->num <= L2->num) {head = tail = L1; L1=L1->next;}
        else {head = tail = L2; L2=L2->next;}  // Assuming both lists have at least 1 item each

        while (1) {
                if (L1 == NULL) {tail->next = L2; break;}
                else if (L2 == NULL) {tail->next = L1; break;}
                if (L1->num <= L2->num) {tail->next = L1; L1=L1->next;}
                else {tail->next = L2; L2=L2->next;}
                tail = tail->next;
        }
return(head);
}
```

Run: endsem-linklist.cpp

Marking Scheme:
1 Mark – Statement (or assumption) regarding structure of a node, similar to struct above.
1 Mark – Parameter passing to and from function merge().
1 Mark – Initialization of the list to be returned.
1 Mark – Loop setup for traversing the lists.
2 Marks – Handling of reaching end of one list before other (L1 becoming NULL before L2, and vice versa).
3 Marks – Rearrangement of pointers for merging, when both L1 and L2 are non NULL.
1 Mark – Return statement.

Note: If a student has
(a) used a dummy node to keep track of the head of the merged list, it is fine.
(b) created new nodes for the merged list, instead of re-using nodes from L1 and L2, deduct 2 Marks.
(c) converted from lists into an arrays, merged the arrays, and converted back to list, deduct 4 Marks.

For any other approach, check logic and give marks accordingly.
If only pseudo-code is given, evaluate the question out of 5 Marks.
Partial marking as per TA discretion.

Write your Roll Number here: ___________________

Q4a: Recall (from Lab 11) that a Stack is a Last-in-First-out (LIFO) structure. Implement a Stack class that provides the methods shown below. (No partial marks for writing only psuedo-code.) **[10 Marks]**

```
class Stack {
  private:
   int store[MAX]; // Stack can hold MAX number of items. Assume each item is a postive integer.
   int nItems; // Number of items currently in the stack.

  public:
   Stack(); // Constructor - initializes variables when a stack object is created.
   bool push(int); // Pushes an item onto the top of the stack, returns false if stack is full.
   int pop(); // Removes (and returns) the item that is currently at the top the stack, returns -1 if stack is empty.
   void top(); // Prints the item at the top of the stack, without removing it.
   int status(); // Returns the number of items currently in the stack.
};
```

```
Stack::Stack() {
        nItems = 0;
}

bool Stack::push (int value) {
      if(nItems == MAX) return false; // stack is full
      store[nItems] = value;
      nItems++; return true;
}

int Stack::pop() {
      if(nItems == 0) return -1; // stack is empty
      nItems--;
      return store[nItems];
}

void Stack::top() {
      if(nItems == 0) cout << "Stack is empty \n";
          else cout << "Item at top of stack is: " << store[nItems-1] << endl;
}

int Stack::status() { return nItems; }
```

Run: endsem-stack.cpp

Marking Scheme:
2 Marks – Constructor
3 Marks – push() - 1 Mark for checking stack full, 1 Mark for inserting item, 1 Mark for updating count.
3 Marks – pop() - 1 Mark for checking stack empty, 1 Mark for removing item, 1 Mark for updating count.
1 Mark – top()
1 Mark – status()

For any other approach, check logic and give marks accordingly.
If only pseudo-code is given, evaluate the question out of 5 Marks.
Partial marking as per TA discretion.

Q4b: A company has three levels of employees – Workers, Managers, and Directors. The annual increment for each level is 20%, 10% and 5%, respectively. Employees may join at different starting salaries and their current salary is calculated by considering their level and number of year's of service. For example, a Worker who joined at salary is X, and has completed one year in the company, will have a current salary of 1.2X. Another Worker who joined at salary Y, and has completed two years, will have a current salary of 1.44Y. Write a class Employee that stores information about an employee and provides functions that can be called from the main program as shown below. **[10 Marks]**

```
#include <iostream>
#include <string>
#include <math.h>
using namespace std;

class Employee {
  private:
        string name;
        int currentLevel; // Assume 1 for Worker, 2 for Manager, 3 for Director
        int yearsCompleted;
        float joiningSalary;
  public:
        // You need to identify the functions required here to make the main program work
        // You may directly write your function defintions immediately after the main program below
};

int main() {
  Employee empA;
  empA.setName("personA");
  empA.setYears(5);  //personA joined 5 years ago (completed 5 years in the company)
  empA.setLevel(1); // personA is a Worker
  empA.setSalary(10); //personA joined at a salary of 10
  float x = empA.getCurrentSalary();

  Employee empB("personB", 3, 2, 20); // personB joined three years ago as a Manager at a salary of 20
  float y = empB.getCurrentSalary();

  if (x > y) cout << empA.getName() << " current Salary is greater than " << empB.getName() << endl;
  else cout << empB.getName() << " current Salary is greater than " << empA.getName() << endl;
return 0;
}

Employee::Employee() { name = " "; currentLevel = 0; yearsCompleted = 0; joiningSalary = 0; }
Employee::Employee(string str, int l, int y, float s)
        { name = str; currentLevel = l; yearsCompleted = y; joiningSalary = s; }

void Employee::setName(string str) { name = str; }
void Employee::setLevel(int l) { currentLevel = l; }
void Employee::setYears(int y) { yearsCompleted = y; }
void Employee::setSalary(float sal) { joiningSalary = sal; }
string Employee::getName() { return name; }

float Employee::getCurrentSalary() {
        float ctc;
        switch (currentLevel) {  // Compute salary based on the level, years completed and joining salary
                case 1: ctc = pow(1.2, yearsCompleted) * joiningSalary; break;
                case 2: ctc = pow(1.1, yearsCompleted) * joiningSalary; break;
                case 3: ctc = pow(1.05, yearsCompleted) * joiningSalary; break;
                default: cout << "Invalid level \n";
        } return ctc;
}
```

Run: endsem-employees.cpp

Marking Scheme: 3 Marks for getCurrentSalary(), and 1 Mark for each of the other functions above.
If only pseudo-code is given, evaluate the question out of 5 Marks.
Partial marking as per TA discretion.