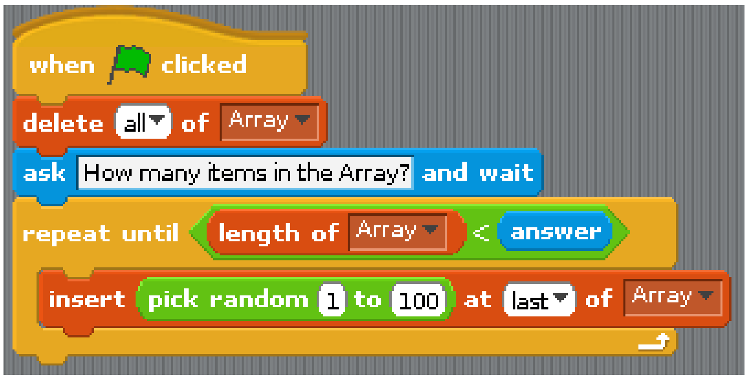
Indian Institute of Technology Bombay, Mumbai
Department of Computer Science and Engineering
CS 101: Computer Programming - Midsem: 23 Feb 2013, 08:30 – 10:30
One sheet of handwritten notes permitted. No photocopies. Weightage: 25%, Max Marks – 50

NOTE: Answer in the space provided. One extra page for rough work is attached at the end. More 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	2c	3a	3b	3c	Total
Marks										

Q1a: Your friend wrote the Scratch program below to initialize an Array with random values in 1-100 but it is not working correctly. Fix the bug(s). **[2 Marks]**



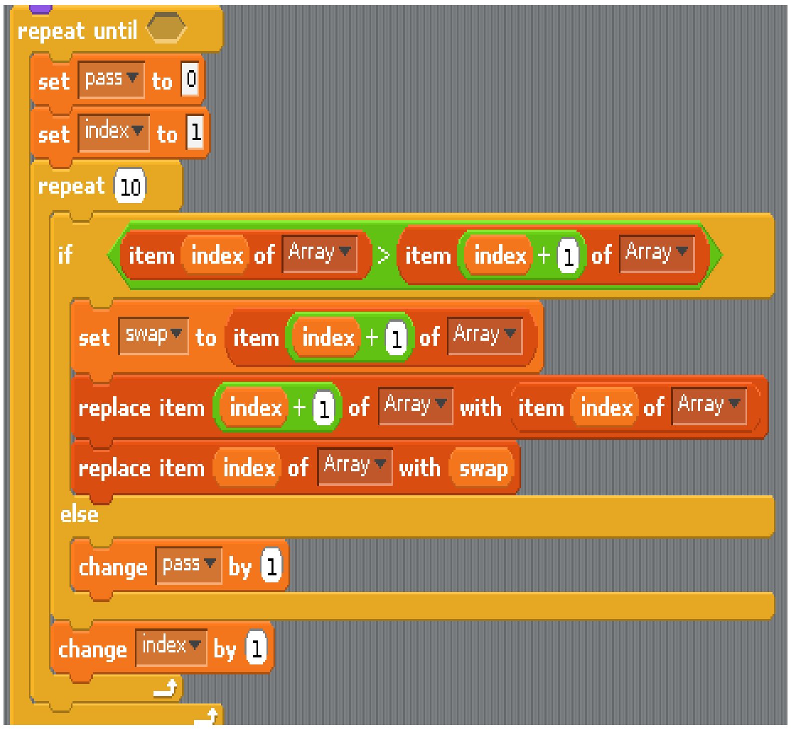
```
when [flag] clicked
delete all of Array
ask How many items in the Array? and wait
repeat until <(length of Array) < (answer)>
    insert (pick random 1 to 100) at last of Array
```

Your Answers:

length of array = max

If ans is as above, then 2 Marks
else if ans is (length of array > max) then 1 Mark, else 0 Marks.

Q1b: After you fixed the bug(s), your friend extended the program as shown below, to sort the Array. Now you have to come up with the correct exit conditions of the 'repeat' loops for the sorting to work. **[4 Marks]**



```
repeat until < >
    set pass to 0
    set index to 1
    repeat 10
        if <(item index of Array) > (item (index + 1) of Array)>
            set swap to item (index + 1) of Array
            replace item (index + 1) of Array with item index of Array
            replace item index of Array with swap
        else
            change pass by 1
        change index by 1
```

Your Answers:

repeat until (pass = length of array)

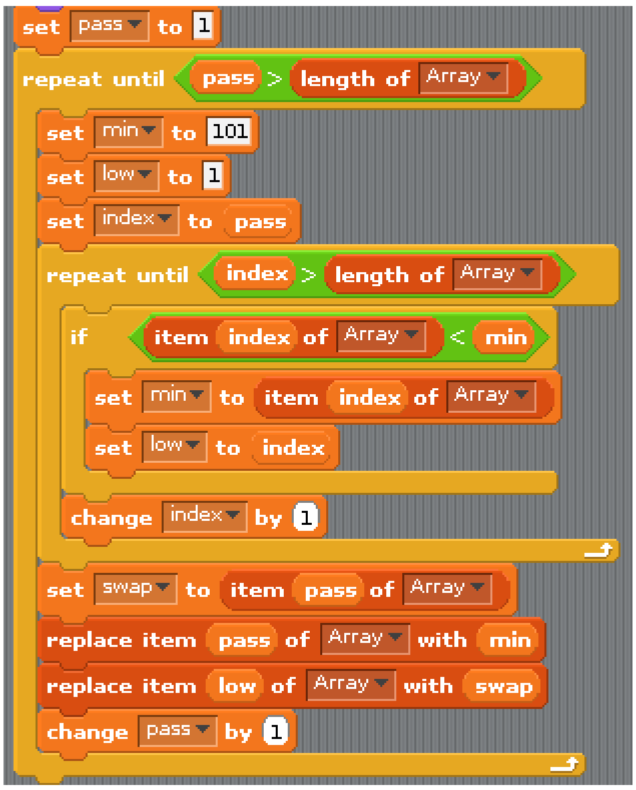
repeat (length of array)

If ans is both of above, then 4 Marks.
If one of the above, then 2 Marks.

Note to TAs:
If some other conditions are given, or the 'repeat' loop is changed to 'repeat until', then check logic for boundary values and give marks accordingly.

Q1c: The program shown in Q1b does Bubble Sort. You need to modify it to Exchange Sort. The C++ version of Exchange Sort was discussed in class. For those who missed that class, here is the basic idea: Find the smallest element in the array and swap it with the first element. Then find the next smallest element and swap it with the second element, and so on. Repeat such a process appropriately till the array is sorted.

Write your Scratch program for Exchange Sort below. **[4 Marks]**



```
set pass to 1
repeat until pass > length of Array
    set min to 101
    set low to 1
    set index to pass
    repeat until index > length of Array
        if item index of Array < min
            set min to item index of Array
            set low to index
        change index by 1
    set swap to item pass of Array
    replace item pass of Array with min
    replace item low of Array with swap
    change pass by 1
```

Use bug-fixed ans from 1a and then the code shown alongside.

Run midsem-exchangeSort-soln.sb to see working of full program.

Outer loop setup is correct – 1 Mark
(initialize pass, check condition, pass++)

Inner loop setup is correct – 1 Mark
(initialize index to pass, check condition, index++)

If block within inner loop is correct – 1 Mark

Setting values of min and low is correct – 0.5 Marks
Swap of values after inner loop is correct – 0.5 Mark

Note to TAs:
If some other logic is given, check if it works like exchange sort and give marks appropriately.

Q2a: Hand-execute the following program and show the contents of the memory locations at the specified steps. **[3 Marks]**

```
int main () {
int num[4]; int* p;

  for (int i=0; i < 4; i++) num[i] = 0;
  p = num;   // Show the memory here ... Step 1

  *p = 10;
  p++;  *(p++) = 20;
  *p = 30;    // Show the memory here ... Step 2

  p = &num[3];  *p = 40;
  *(p-1) = 30;

  for (int n=0; n<4; n++)
    cout << num[n] << ", ";
  return 0;
} // Show the final output  ... Step 3
```

Step1: Contents of the memory locations are:

num[0] has memory byte address X, value 0.
num[1] has address X+4, value 0, and so on.
P has address Y, value X. Or P should be drawn as pointing to num[0]. Note: Y could be X+16.

All items are shown correctly – 1 Mark
Items are shown correctly but some details are missing – 0.5 Marks.
Anything else – 0 Marks.

Step 2: Contents of the memory locations are:
num[0] = 10, num[1] = 20, num[2] = 30,
p pointing to num[2].

All values are correct – 1 Mark. Else 0 Marks.

Step 3: The output of the program is:
10, 20, 30, 40

All values are correct – 1 Mark. Else 0 Marks.

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

Q2b: Recall the class discussion on reversing a number. The program below takes an integer *a* as input and outputs its reverse integer *b.* Example: if a = 12345, then b is 54321. The program does not use Arrays. One or more statements in the *while loop* got deleted accidentally. Fill in the missing statement(s). **[3 Marks]**

```
#include <iostream>
using namespace std;
int main() {
  int a, b = 0;
  cin >> a;
    while (a > 0) { // ... Write your answer below ...
b = a % 10;
    a /= 10;
    cout << b;
// Since cout << b is inside the while loop, the solution above is
ok, even though it prints each digit of b as independent numbers.
  }
  return 0;
}
```

Space for Rough Work:

b = a % 10; - 3 Marks

OR

b = b * 10 + a % 10; and move cout << b; to outside the while loop – 3 Marks

Note to TAs:
If there are other statements, but final answer is correct, give 2 Marks.
Wrong answer - 0 Marks.

Q2c: You are writing the number-guessing game shown below. It generates an array of random integers and asks the user to guess a *number*. If *number* is part of the array, the user wins and the program terminates, else the user can play again. Complete the missing portions of the program. **[4 Marks]**

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

int main() {
 int n = 9, arr[n], i; char flag = 'y';
 srand (time (NULL));
 while ( flag != 'n'      ) { // ... Fill in the Condition here ...
// generate a random number in [0..99] and insert it into array
  for (i = 0; i < n; i++)
    arr[i] =  rand() % 100;   // ... Fill in the Statement here ...

// get the input
    int num;
    cout << "Give your number: "; cin >> num;

// Check if input is in the array ... Fill in the code below ...
    for (i = 0; i < n; i++) {
     if (arr[i] == num) {
       cout << "You Win! The number is at position: " << i << endl;
       return 0;
     }
    }

// print the array
  cout << "Sorry, the numbers were: ";
  for (i = 0; i < n; ++i) cout << arr[i] << ", ";
  cout << endl << "Play again (y/n)?: "; cin >> flag;
 }
 return 0;
}
```

Space for Rough Work:

Compile midsem-numGame.cpp and run it to see working of program.

(flag != 'n') -- 1 Mark. Else 0 Marks.

rand() % 100; -- 1 Mark. Else 0 Marks.

For loop correct – 2 Marks. Else 0 Marks.

Q3a: Write a C++ program that given an integer *n*, generates a sequence R as follows:
R_1 = product of the digits of *n*, R_2 = product of the digits of R_1, and so on, till some R_i becomes a single digit.

For example, if n = 999, then R_1 = 729, R_2 = 126, R_3 = 12, and R_4 = 2. So your program should output 729, 126, 12, 2, and terminate. If n = 6725, your program should output 420, 0, and terminate. **[10 Marks]**

Compile and run midsem-numSequence.cpp, to see working. Code shown below.

```
#include <iostream>
using namespace std;
int main() {
int num, d[10], newN;
int size, i;

cout << "Give a number (upto 10 digits): "; cin >> num;
do {
        cout << "The number is: " << num << endl;
        i = 0;
        while (num > 0) {
                d[i] = num % 10; i++;
                num /= 10;
        }
        size = i;
        newN = 1;
        for (i = 0; i < size; i++) newN = newN * d[i];
        cout << "The product of its digits is: " << newN << endl;
        num = newN;
} while (newN/10 > 0);
return 0;
}
```

Check the program/pseudo-code logic for the following key steps:

1. Read the number - 1 Mark. It is ok to assume that the user will not exceed the specified number of digits. If someone has done a check and reported error, that is good.
2. Loop to extract the digits into an array - 2 Marks.
3. Loop to compute the product of the digits into newN - 2 Marks.
4. Output newN – 1 Mark.
5. Loop to repeat steps 2-4 till newN becomes single digit – 2 Marks.

If all the logic is correct AND C++ syntax is used – [Plus 2 Marks to above].
If all the logic is correct BUT only pseudo-code is given – [Minus 2 Marks to above].

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

Q3b: Definition of Perrin numbers is: P(0) = 3, P(1) = 0, P(2) = 2, and P(n) = P(n − 2) + P(n − 3) for n > 2. Write a C++ program that takes as two numbers *a* and *b* as input, and counts how many perrin numbers are odd and how many are even, in the range [a,b] (including a and b). **[10 Marks]**

Compile and run midsem-perrinNum.cpp, to see working. Code shown below.

```
#include<iostream>
using namespace std;
int main(){
int perrin[50]={3,0,2};
int lower_range, upper_range, index=3, count_even=0, count_odd=0;

cout << "Enter lower and upper range. Lower range should be greater than 3.\n";
cin >> lower_range >> upper_range;
cout << endl;

if(lower_range > 3 && lower_range < upper_range) {
  perrin[index] = perrin[index-2] + perrin[index-3];
  while(perrin[index] <= upper_range) {
          if(perrin[index] >= lower_range) {
                  cout << perrin[index] << "\n";
                  if(perrin[index]%2 == 0) count_even++;
                  else count_odd++;
   }
   index++;
   perrin[index] = perrin[index-2] + perrin[index-3];
  }
  cout<<"\n Even perrin Numbers : "<< count_even;
  cout <<"\n Odd perrin Numbers : "<<count_odd<<"\n";
}
else cout<<"Range given is not correct";
return 0;
}
```

Check the program/pseudo-code logic for the following key steps:

1. Read the input and check validity (if condition) - 1 Mark.
2. Initialize perrin array and compute perrin[index] as per formula – 1 Mark.
3. Loop to compute successive perrin[index] values - 2 Marks.
4. If condition to check when to start counting even and odd - 2 Marks.
5. Correct counting of even and odd – 1 Mark.
6. Output – 1 Mark.

If all the logic is correct AND C++ syntax is used – [Plus 2 Marks to above].
If all the logic is correct BUT only pseudo-code is given – [Minus 2 Marks to above].

Q3c: Consider Pascal's triangle shown below. Let the rows be numbered sequentially, with the top row being Row Number 0. Observe that: (i) each number is the sum of the two numbers above it, and (ii) the numbers in each row are *binomial coefficients*, i.e., the numbers in row *n* are coefficients in the expansion of $(1 + x)^n$. Write a program to take Row Number as input and output the numbers in that Row. **[10 Marks]**

Extra Credit [3 Marks]: Output the triangle upto the given Row, with appropriate number of spaces to get a visual formatting as shown.

```
           1
         1   1
       1   2   1
     1   3   3   1
   1   4   6   4   1
 1   5  10  10   5   1
```

Compile and run midsem-pascalTriangle.cpp to see working program.
Code shown below.

```
#include <iostream>
using namespace std;
int main() {
int n,k,i,x;
cout << "Enter a row number for Pascal's Triangle: "; cin >> n;
// --------------------------------- version 1
// To output only the values of the given row
        x=1;
        for(k=0;k<=n;k++) {
                cout << x << " ";
                x = x * (n - k) / (k + 1);
        }
        cout << "\n\n";
// --------------------------------- version 2
// To simply output all the rows upto the given row, without formatting
  for(i=0;i<=n;i++) {
        x=1;
        for(k=0;k<=i;k++) {
                cout << x << " ";
                x = x * (i - k) / (k + 1);
        }
        cout << endl;
  }
//------------------------------ version 3
// To output all the rows upto the given row, with formatting
  cout << endl;
  for(i=0;i<=n;i++) {
// Insert initial spaces according to each row.
// RHS of output will be slightly skewed after row 5, due to 2 digit numbers, but that is ok.
        for (int j=0; j<=(n-i); j++) cout << " ";
        x=1;
        for(k=0;k<=i;k++) {
                cout << x << " ";
                x = x * (i - k) / (k + 1);
        }
        cout << endl;
  }
return 0;
}
```

If solution given is like version 1 above – only outputs values of the given row – 6 Marks.
If solution is like version 2 - outputs values of all the rows upto the given row – 8 Marks.
If solution includes adding spaces for formatting all the rows upto the given row – 10 Marks.

If all the logic is correct AND C++ syntax is used – [Plus 3 Marks to above].
If all the logic is correct BUT only pseudo-code is given – [Minus 3 Marks to above].

If some other logic is used, check the formula and give marks accordingly, similar to above.