**CS101 INTRODUCTION TO COMPUTING**

**FINAL, FALL 2014**

**Date: 22thDec, 2014 Max. Marks:60 Time: 180 minutes**

**IMPORTANT:** Solve the exam in the space provided. Write your name, roll number and section on every page. You can ask for extra sheets for rough work but any extra sheets ***WILL NOT*** be marked or collected. Any assumptions must be written clearly in comments. **Good luck!**

**QUESTION 1**

a. What is the output of the following program?

char str[] = { 'a','0',0, '1', '5', '0',0};

cout<<str;

a0

b. The following code should sort an array in an ascending (increasing) order. However, it contains some logical errors (no syntax error). Your job is to identify & correct all logical errors in it USING MINIMAL MODIFICATIONS. The input and the expected output of code are also given.

**Input**: num = {8,3,1,2,4,7,5,6}, length = 8

**Output**:num = {1, 2, 3, 4, 5, 6, 7, 8}

|  |  |
| --- | --- |
| **Code** | **Correction (if any)** |
| void SortAscending(int num[], int Length) |  |
| { | //1st loop is currently sorting in descending order |
| int i, j, first, temp; | //also variable ‘first’ and ‘j’ should be given |
| first = 0; | //value of ‘i’. |
| for ( i = Length - 1; i > 0; i--) { | //correct version is: |
| for ( j = 1; j <= Length; j++) { | for ( i = 0; i < Length; i++) { |
| if ( num[j] < num[first] ) | first = i; |
| first = j; | for( j=i ; j < Length; j++){ |
| } |  |
| num[first] = num[i]; | //putting variable in temp must come first |
| temp = num[first]; | temp = num[first]; |
| num[i] = temp; | num[first] = num[i]; |
| } | num[i] = temp; |
| } |  |

c. Convert FED from hexadecimal notation to binary

F=1111, E=1110, D=1101

FED = (111111101101)2

d. Convert 01110011 from binary to hexadecimal

0111 = 7, 0011 = 3

01110011 = (73)16

**QUESTION 2**

The number *n* is evil if it has an even number of ones in its binary expansion. For example, 1032 is evil because its binary expansion 0000 0100 0000 1000 has two ones. However, 1033 is not evil. Write a C++ function to determine whether or not a number is evil. You may assume *n* will always be an integer.

bool evilOrNot(int n)

{

int ones = 0;

for (; n > 0; n /= 2)

{

if (n % 2 == 1)

ones++;

}

if (ones % 2 == 0)

return true;

return false;

}

**QUESTION 3**

What is the output of the following program?

|  |  |
| --- | --- |
| bool check(int&number)  {  int count=0;  bool answer=false;  for(int a=1;a<=number;a++)  {  if(number%a==0)  {  count++;  }  }  if(count==2)  {  answer=true;  }  else  {  answer=false;  }  number++;  return answer;  } | void printSomething(int number)  {  for(inti=number; i>=1; i--)  {  for (int k= number;k>i; k--)  {  cout<<"O";  }  int j = 1;  while(j<=i)  {  if(check(j))  cout<<"A";  else  cout<<"H";  }  cout<<endl;  }  }  int main()  {  int size=5;  printSomething(size);  } |

HAAHA

OHAAH

OOHAA

OOOHA

OOOOA

**QUESTION 4**

Implement the C++ function **FillWithMean** that takes two arrays, **in** and **out**, and a positive number **N**as parameters. The functionthen fillseach index **i**of the array **out**with the mean/average of**N\*2+1** values in the neighbourhood of **i**from the array **in**.That is, it will take the mean of the ithnumber and **N** numbers to its left and **N** numbers to its right. You also need to take care of the boundaries as shown in the examples below.

EXAMPLE 1 with N=1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **index** | **0** | **1** | **2** | **3** | **4** | **5** |
| **in** | 4 | 5 | 3 | 6 | 1 | 3 |
| **out** | mean(4,5)  =9/2 | mean(4,5,3)  =4 | mean(5,3,6)  =14/3 | mean(3,6,1)  =10/3 | mean(6,1,3)  =10/3 | mean(1,3)  =2 |

EXAMPLE 2 with N=2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **index** | **0** | **1** | **2** | **3** | **4** | **5** |
| **in** | 4 | 5 | 3 | 6 | 1 | 3 |
| **out** | mean(4,5,3)  =4 | mean(4,5,3,6)  =18/4 | mean(4,5,3,6,1)  =19/5 | mean(5,3,6,1,3)  =18/5 | mean(3,6,1,3)  =13/4 | mean(6,1,3)  =10/3 |

Solution:

void FillWithMean (float in[], float out[], int N, int length)

{

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

{

float sum = 0, num = 0, mean = 0;

for (int j = i; j <= (i + N) && j < length; j++)

{

sum += in[j];

num++;

}

for (int j = (i - 1); j >= (i - N) && j >= 0; j--)

{

sum += in[j];

num++;

}

mean = sum / num;

out[i] = mean;

}

}

**QUESTION 5**

A string is a palindrome if it remains unchanged when reversed. For example “abba” and “bazab” are palindromes.Implement a C++ function **isSpacedPalindrome** which determines whether or not a string is a palindrome when we ignore all spaces. For example, **"race car"** and **"a b c d dcba"**are spaced palindromes.

bool isSpacedPalindrome(char str[], int size)

{

char temp[100], temp2[100];

int j = 0, k = size - 1;

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

{

if (str[i] != ' ')

temp[j++] = str[i];

}

temp[j] = '\0';

int n=0;

while (temp[n++] != '\0');

for (int i = 0, k = (n - 2); i < (n-1); i++)

temp2[k--] = temp[i];

temp2[n] = '\0';

for (int x = 0; x < (n - 1); x++)

{

if (temp[x] != temp2[x])

return false;

}

return true;

}

**QUESTION 6**

Write a C++ function **findString** which finds a given string**str** in a given 2-dimensional array of upper-cased characters with wrap-around. The function should returnas parameters the starting row and column of the first occurrence of the string to be searched if it is present, and should return false otherwise.It should take the following parameters as input

1. A two-dimensional array of characters. The number of columns is fixed at 20.
2. Number of rows in the array.
3. A null terminated string **str**that needs to be searched. **str**can be one or more characters long.

You must implement**at least two** other functions that the**findString**function calls to complete the task. Example outputs where **str= "CAT"** are given below.In wrap around, if we reach the end of the row, we continue from the start of the SAME row.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Output = **2 1** | | | | | | | | | C | C | S | H | J | U | E | B | | L | M | N | C | A | L | C | A | | D | **C** | **A** | **T** | J | S | W | T | | Forward | | | | | | | | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Output = **1 6** | | | | | | | | | C | C | S | H | J | U | E | B | | **T** | M | N | C | A | L | **C** | **A** | | D | C | A | P | J | S | W | T | | Forward wrap around | | | | | | | | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Output = **2 3** | | | | | | | | | C | C | S | H | J | U | E | B | | S | M | N | C | A | L | C | A | | D | **T** | **A** | **C** | J | S | W | T | | Backward | | | | | | | | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Output = **0 0** | | | | | | | | | **C** | C | S | H | J | U | **T** | **A** | | M | M | N | C | A | T | C | A | | D | I | J | U | J | S | W | T | | Backward wrap around | | | | | | | | |

bool checkWord(char arr[][20], char str[], int idir, int i, int j)

{

int colsize = 0;

while (arr[0][colsize++] != '\0');

colsize--;

//idir is direction index

int DIR[2] = { 1, -1 }, k = 0;

for (int col = j; col < colsize && col >= 0 && str[k]!='\0'; col += DIR[idir])

{

if (toupper(str[k++]) != arr[i][col])

return false;

if (str[k] != '\0' && col == 0 && idir == 1)

col = colsize;

else if (str[k] != '\0' && col == (colsize-1) && idir == 0)

col = -1;

}

return true;

}

bool checkPresence(char a, char b)

{

if (a == b)

return true;

return false;

}

bool findString(char arr[][20], int rowsize, char str[], int &sr, int &sc)

{

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

{

for (int j = 0; j < arr[i][j] != '\0'; j++)

{

if (checkPresence(arr[i][j], toupper(str[0])))

{

for (int idir = 0; idir < 2; idir++)

{

if (checkWord(arr, str, idir, i, j))

{

sr = i, sc = j;

return true;

}

}

}

}

}

return false;

}