Fast**National University of Computer & Emerging Sciences, Karachi  
Fall-2019Computer Science / Software Engineering Department  
Final Examination  
10thDec 2019, 09:00 am – 12:00 noon**

|  |  |  |
| --- | --- | --- |
| **Course Code: CS118** | **Course Name:Programming Fundamentals** | |
| **Instructor Name: Dr. Farooque/ M. Shahzad/Shoaib Rauf/ BasitJasani** | | |
| **Student Roll No:** | | **Section No:** |

Instructions:

* Return the question paper and make sure to keep it inside your answer sheet.
* Read each question completely before answering it. There are **8questions and 3pages**.
* In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.
* You are **not allowed to write** anything on the question paper (except your ID and group).

**Time**: 180 minutes. **Max Marks**: 124

1. Observe and try to understand the following programs. There are no syntax errors in the programs. Write errors (runtime/compile time) with reasons if there are any available or write outputs if the programs are fine. **[Marks 3\*4=12, 10 minutes]**

|  |  |
| --- | --- |
| (i)  char\* func (char \*ptr)  {  ptr+=8;  return ptr;  }  main()  {  char \*x, \*y;  x="Programming Fundamentals";  y=func(x);  printf(" y = %s", y);  }  **Y= ing Fundamentals** | (ii)  void fun(const int \*ptr)  {  \*ptr = 30;  }  main()  {  int y = 40;  fun(&y);  printf("%d", y);  }  **Compile Time Error** |
| (iii)  main()  {  int ary[2][2][3] = {  {  {1,2,3},  {4,5,6}},  {{7,8,9},  {10,11,12}  }  };  int \*p; p = &ary;  printf("%d %d",\*p, \*p+11);  }  **1 12** | (iv)  main()  {  char \*arr[] = { "ant", "bat", "cat", "dog", "egg", "fly" };  function(arr);  }  void function(char \*\*ptr)  {  char \*ptr1;  ptr1 = (ptr += sizeof(int))[-2];  printf("%s\n", ptr1);  }  **cat** |

1. Choose the most appropriate answer from given choices. Write answer in the answer sheet. **[Marks 3\*4=12, 10 minutes]**

|  |  |
| --- | --- |
| (i)  If x is anone-dimensional array, then  A. &x [i] is same as x + i - 1  B. \* (x + i ) is same as \* (&x [i] )  C. \* (x + i) is same as x[i]  **D. both (b) & (c)** | (ii)  Which of the following comments about arrays and pointers is/are not true?  A. Both are exactly same  B. Array is a non-constant pointer  C. Pointer is a1D and dynamic array  **D. All of these** |
| (iii)  Consider the following program fragment.What will be the output?  static char wer[3][4] = {"bag", "let", "bud"};  putchar (\* (wer [1] + 1)) ;  **A. e**  B. a  C. 1  D. b | (iv)  How can you write a[i][j][k] in an equivalent pointer expression?  A. ((\*\*\*(a+i)+j)+k)  B. (\*\*(\*(a+i)+j)+k)  C. (\*(\*(a+i)+j)+k)  **D. \*(\*(\*(a+i)+j)+k)** |

1. Using nested loops, produce given output format: **[Marks 5\*2=10, 30 minutes]**
2. **Output sequence:**10, 13, 26, 29, 58, 61
3. Write a ‘C’ program that reads a string in dynamic array. Then draw the pyramid pattern of a user provided string.

**Sample Input: Zeshan**

**Sample Output:**



(a)

**int main()**

**{**

**int x =10,i;**

**printf("%d",x);**

**for (i=0; i<10 ;i++)**

**{**

**if( (i%2) == 0 )**

**{**

**x+=3;**

**printf(", %d",x);**

**}**

**else**

**{**

**x\*=2;**

**printf(", %d",x);**

**}**

**}**

**return 0;**

**}**

**(b)**

**int main()**

**{**

**int n,i=0,j=0;**

**printf("\nEnter no of characters you want to enter : ");**

**scanf("%d",&n);**

**char \*ptr=(char \*)malloc(n\*sizeof(char));**

**printf("\nEnter the string: ");**

**scanf("%s",ptr);**

**printf("\n\n");**

**for( ;i<n ; i++)**

**{**

**for(j=0; j<=(n-i); j++)**

**{**

**printf(" ");**

**}**

**for(j=0; j<=i; j++)**

**{**

**printf("%c ",ptr[j]);**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

1. Ali has stored prices of all his sold products in linear 1D dynamic array. Now, he wants you to develop a single function which can calculate sum of all the products on odd indexes and even indexes separately using recursion.  **[Marks 15, 20 minutes]**

**#include <stdio.h>**

**int sumOfEvenOdd(int start, int end);**

**int main()**

**{**

**int start, end, sum;**

**/\* Input lower and upper limit from user \*/**

**printf("Enter lower limit: ");**

**scanf("%d", &start);**

**printf("Enter upper limit: ");**

**scanf("%d", &end);**

**printf("Sum of even/odd numbers between %d to %d = %d\n", start, end, sumOfEvenOdd(start, end));**

**return 0;**

**}**

**int sumOfEvenOdd(int start, int end)**

**{**

**/\* Base condition \*/**

**if(start > end)**

**return 0;**

**else**

**return (start + sumOfEvenOdd(start + 2, end));**

**}**

1. Junaid wants to store all his customer data in an encrypted text file. Help him by developing a program which contains two functions that can encrypt and decrypt the file (data.txt). Make sure that the encrypted file text is completely unreadable.Encryption means that we want to change text present in a file to look like something else. For example, an encrypted text “abcmno” can look like “cdeopq”. In decryption, we will get original text back from encrypted text. **[Marks 10+10=20, 25 minutes]**

**Void fun1()**

**{**

**fptt=fopen("temp.txt", "w");**

**if(fptt==NULL)**

**{**

**printf(" Error in creation of file temp.txt ..!!");**

**fclose(fpts);**

**exit(2);**

**}**

**while(1)**

**{**

**ch=fgetc(fpts);**

**if(ch==EOF)**

**{**

**break;**

**}**

**else**

**{**

**ch=ch+100;**

**fputc(ch, fptt);**

**}**

**}**

**fclose(fpts);**

**}**

**Void fun2()**

**{**

**fptt=fopen("temp.txt", "r");**

**if(fptt==NULL)**

**{**

**printf(" File does not exists or error in opening..!!");**

**fclose(fpts);**

**exit(9);**

**}**

**while(1)**

**{**

**ch=fgetc(fptt);**

**if(ch==EOF)**

**{**

**break;**

**}**

**else**

**{**

**ch=ch-100;**

**fputc(ch, fpts);**

**}**

**}**

**printf(" The file %s decrypted successfully..!!\n\n",fname);**

**fclose(fpts);**

**fclose(fptt);**

**}**

1. Write a 'C' program that have a 2D-array data of 30 players, from where you have to select the batsmen on given certain criteria. **[Marks 10\*2=20, 25 minutes]**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Matches | Runs | Fifty’s | Hundred’s | Four’s | Sixes’ |
| 1 | 34 | 1700 | 1 | 1 | 40 | 5 |

1. Write a function to list ID’s of top 3 scorers who scored at least 1 Hundred or 2 Fifty’s

void Top3MostTon( int Player[][7], int Total\_rows );

1. Write a function to list ID’s of top 3 boundary hitter. [boundary = Sixes or Fours]

void Top3BoundaryHitter( int Player[][7], int Total\_rows);

**Note:** Use **pointer functions** to call above functions

**(a) & (b).**

**#include<stdlib.h>**

**#include<stdio.h>**

**void Top3MostTon( int Player[][7], int Total\_rows);**

**void Top3BoundaryHitter( int Player[][7], int Total\_rows);**

**void swapRow(int Player[][7], int rGreater, int rLower);**

**int main()**

**{**

**int i=0,j=0;**

**int Players[30][7];**

**for(i=0; i< 30; i++)**

**{**

**for(j=0; j < 7; j++)**

**{**

**scanf("%d ",&Players[i][j]);**

**}**

**printf("\n");**

**}**

**void (\*playerPtr[])(int [][7],int) = {Top3MostTon,Top3BoundaryHitter};**

**(playerPtr[0])(Players,30);**

**return 0;**

**}**

**void Top3MostTon(int Player[][7], int Total\_rows)**

**{**

**int i=0,j=0,run=0;**

**run = Player[0][2];**

**for(;i<Total\_rows; i++)**

**{**

**for(j=i+1; j < 7; j++)**

**{**

**if( run< Player[j][2] )**

**{**

**swapRow(Player,j,i);**

**}**

**}**

**}**

**for(i=0;i < 3 ; ++i)**

**{**

**printf("\n Top 3 Player Id");**

**if( (Player[i][4] >= 1 ) || (Player[i][3] >= 2 ) )**

**{**

**printf("\n Player Id: %d",Player[i][0]);**

**}**

**}**

**}**

**void Top3BoundaryHitter( int Player[][7], int Total\_rows)**

**{**

**int i=0,j=0,shot=0;**

**shot = (Player[0][5] + Player[0][6]);**

**for(;i<Total\_rows; i++)**

**{**

**for(j=i+1; j < 7; j++)**

**{**

**if( shot< (Player[j][5] + Player[j][6]) )**

**{**

**swapRow(Player,j,i);**

**}**

**}**

**}**

**printf("\n Top 3 Player Id");**

**for(i=0;i < 3 ; ++i)**

**{**

**printf("\n Player Id: %d",Player[i][0]);**

**}**

**}**

**void swapRow(int Player[][7], int rGreater, int rLower)**

**{**

**int i=0,temp=0;**

**for(i=0 ; i<7; i++)**

**{**

**temp = Player[rLower][i];**

**Player[rLower][i] = Player[rGreater][i];**

**Player[rGreater][i] = temp;**

**}**

**}**

1. Write a 'C' program and consider the following two 2D arrays named as **CellNameData** and **CellExpData**as given below. The **CellNameData** array with 4×2 dimension, contains the cell no. and cell names, whereas **CellExpData** array with 5×6 dimension, contains the 4 types of gene expression values against each cell names. The first row in both array and the first column in **CellExpData** array are just row and column heads. Out of the 5 different columns in **CellExpData** array, select only those columns that match with the cell names given in **CellNameData** array. Store these data into new 2D array named as **MergeData**. After getting selected columns, expected data in **MergeData** array are shown below: **[Marks 15, 25 minutes]**

[Hint: You can assume numeric values to represent the each row and column names. No points will be given on hard-coding except declaration and initialization statements]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **CellNameData** | | | **CellNo.** | **CellName** | | 0.1 | LiverCell | | 0.2 | KidneyCell | | 0.3 | LungCell | | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **CellExpData** | | | | | | | **Gnames** | **LungCell** | **SkinCell** | **IntestineCell** | **LiverCell** | **KidneyCell** | | **Gene1** | 2.3 | 1.7 | 4.3 | 3.4 | 4.5 | | **Gene2** | 1.5 | 3.6 | 8.1 | 5.5 | 2.9 | | **Gene3** | 4.1 | 6.6 | 5.3 | 3.9 | 8.1 | | **Gene4** | 9.9 | 2.7 | 6.2 | 5.8 | 3.3 | |
| |  |  |  |  | | --- | --- | --- | --- | | **MergeData** | | | | |  | **LiverCell** | **KidneyCell** | **LungCell** | | **Gene1** | 3.4 | 4.5 | 2.3 | | **Gene2** | 5.5 | 2.9 | 1.5 | | **Gene3** | 3.9 | 8.1 | 4.1 | | **Gene4** | 5.8 | 3.3 | 9.9 | | |

**#include<stdio.h>**

**#include<string.h>**

**void main(){**

**float LungCell=1.1, SkinCell=1.2, IntestineCell=1.3, LiverCell=1.4, KidneyCell=1.5;**

**float Gnames=0.0, Gene1=0.1, Gene2=0.2, Gene3=0.3, Gene4=0.4;**

**float CellNo=0.0, CellName =0.1;**

**char CellString[6][14]={"Gnames","LungCell", "SkinCell", "IntentineCell", "LiverCell", "KidneyCell"};**

**char tempCell[6][14];**

**float CellExpData[5][6]={{Gnames,LungCell,SkinCell,IntestineCell,LiverCell,KidneyCell},**

**{Gene1,2.3,1.7,4.3,3.4,4.5},**

**{Gene2,1.5,3.6,8.1,5.5,2.9},**

**{Gene3,4.1,6.6,5.3,3.9,8.1},**

**{Gene4,9.9,2.7,6.2,5.8,3.3}};**

**float CellNameData[4][2]={**

**{CellNo, CellName},**

**{0.1, LiverCell},**

**{0.2, KidneyCell},**

**{0.3, LungCell}**

**};**

**float MergeData[5][5];**

**int i, j, k, l=0, t=0, col;**

**for(i=0;i<=2;i++){**

**for(j=0;j<=3;j++){**

**for(k=1;k<=5&&j==0;k++){**

**if(CellNameData[l+1][1]==CellExpData[0][k])**

**{**

**strcpy(tempCell[t],CellString[k]);**

**t++,l++;col=k;break;**

**}**

**}**

**MergeData[j][i]=CellExpData[j+1][col];**

**}**

**}**

**for(i=0;i<t;i++)**

**printf("\t %s",tempCell[i]);**

**printf("\n");**

**for(i=0;i<=3;i++){**

**printf("Gene%d\t",i+1);**

**for(j=0;j<=2;j++)**

**{**

**printf(" %f\t ", MergeData[i][j]);**

**}**

**printf("\n");**

**}**

**}**

1. Assume that the 100 academic journals records are stored in a txt file (data.txt) as shown:

**[Marks 10\*2=20, 25 minutes]**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Title** | **Journal** | **Issue** | **Year** | **startPage** | **endPage** | **Author** |
| 1 | Cuckoo Hashing | JAIg | 51 | 2004 | 121 | 133 | Robert Tarjan |
| 2 | Deterministic Dictionaries | JAIg | 41 | 2001 | 69 | 85 | Robert Lafore |
| .. | .. | .. | .. | .. | .. | .. | .. |
| 100 | What Godel missed | SICO | 51 | 2008 | 1 | 5 | Paul Bailey |

**Hint:** struct Article{int ID; char Title[30]; char Journal[50];….};

**Note:** “No global variables allowed. Use appropriate data types, return types and function arguments if not explicitly mentioned.”

1. Write a function named “Check\_Author(…)”. This function searches the data for all the articles authored by “Robert Tarjan” and changes the author to “Robert Lafore”. If no such articles exist the function must display “Not Found”.

**Check\_Author()**

**{**

**int n;**

**FILE\* data;**

**if ((data = fopen("data.txt", "rb")) == NULL)**

**{**

**printf("Error opening file\n");**

**}**

**struct Article\* a1 = (struct Article\*) malloc(sizeof(struct Article) \* 100);**

**for (n = 0; n<100; n++)**

**{**

**fread(&a1[n], sizeof(struct Article), 1, data);**

**if(strcmp(a1[n].author, "Robert Tarjan")==0)**

**{**

**strcpy(a1[n].author, "Robert Lafore");**

**//printf("\n%s", a1[n].author);**

**}**

**}**

**fclose(data);**

**if ( (data = fopen("data.txt", "wb")) == NULL )**

**{**

**printf("Error opening file\n");**

**}**

**fwrite(a1, sizeof(struct Article) \* 100, 1, data);**

**fclose(data);**

**free(a1);**

**}**

1. Write a function named “Article\_Count(…)”. This function displays all the records for which the totals pages are 50 or above and year 2000 or above.

**Article\_Count()**

**{**

**int n;**

**FILE\* data;**

**if ((data = fopen("data.txt", "rb")) == NULL)**

**{**

**printf("Error opening file\n");**

**return 1;**

**}**

**struct Article\* a1 = (struct Article\*) malloc(sizeof(struct Article) \* 100);**

**int count = 0, avg = 0;**

**for (n = 0; n<100; n++)**

**{**

**fread(&a1[n], sizeof(struct Article), 1, data);**

**if((a1[n].endPage - a1[n].startPage)>=50 && a1[n].year>=2000 )**

**{**

**printf("\n%d", a1[n].id);**

**printf("\n%s", a1[n].title);**

**printf("\n%s", a1[n].journal);**

**printf("\n%d", a1[n].issue);**

**printf("\n%d", a1[n].year);**

**printf("\n%d", a1[n].startPage);**

**printf("\n%d", a1[n].endPage);**

**printf("\n%s", a1[n].author);**

**}**

**}**

**}**

***BEST OF LUCK!***