**King Fahd University of Petroleum and Minerals**

**Information and Computer Science Department**

ICS 103: Computer Programming in C

**Summer Semester 2010-2011 (Term-103)**

##### Major Exam-II

**Time:120 minutes Wednesday, August 3, 2011**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name:** |  | | | | | | | | | | |
| **ID#:** |  |  |  |  |  |  |  |  |  |  |  |

PLEASE CIRCLE YOUR SECTION BELOW:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Section | 01 | 02 | 03 | 04 | 05 |
| Instructor | Dr. Aiman | Dr. Aiman | Mr. Faisal | Mr. Putu | Dr. Azzedin |

|  |  |  |
| --- | --- | --- |
| Question # | Maximum Marks | Obtained Marks |
| 1: P1,P2,P3 | 8+7+6 |  |
| 1: P4,P5,P6 | 6+9+6 |  |
| 2 | 16 |  |
| 3 | 7 |  |
| 4 | 18 |  |
| 5 | 17 |  |
| Total | 100 |  |

**Notes:** 1. Make sure you have **NINE** pages including the cover page.

2. Closed book and notes.

3. No calculator is allowed.

4. Write clearly, briefly and precisely.

5. Turn off all your communication gadgets.

6. Cheating will result in ZERO grade.

## Good Luck

## Question 1: (42 points)

|  |  |
| --- | --- |
| #include <stdio.h> //**P1: 8 points**  int main()  {  int num[10]={5,-2,3,1,1,0,2,-4,3};  int i, k, sum;  float avg, j;  sum=0; avg=0; j=0; k=0;  for(i=0; i<10; i++)  {  if (num[i]>=0)  { sum += num[i];  j=j+1;  } else {  k=k+1;  }  }  printf("Total=%d\n",sum);  avg=sum/j;  printf("Average=%.2f\n",avg);  printf("k=%d\n",k);  return 0;  } | **Total=15**  **Average=1.88**  **k=2** |
| #include <stdio.h> // **P2: 7 points**  int main(void)  {  int i, j, x=1;  for(i=1; i<5; ++i)  for(j=1; j<i; ++j)  {  x += (i+j-1);  printf("%d\n", x);  }  printf("x=%d", x );  return 0;  } | **3**  **6**  **10**  **14**  **19**  **25**  **x=25** | |

|  |  |
| --- | --- |
| #include <stdio.h> // **P3: 6 points**  void fun1(int n);  void fun2(int n);  int main(void){  fun2(3);  return 0;  }  void fun1(int n){  for (int i=n; i>0; i--)  printf("%d",i);  printf("\n");  }  void fun2(int n){  if (n==0) return;  else {  fun1(n);  fun2(n-1);  fun1(n);  }  } | **321**  **21**  **1**  **1**  **21**  **321** |

|  |  |
| --- | --- |
| #include <stdio.h> // **P4: 6 points**  int main(void)  {  int x = 25, y = 0;  int \*p1, \*p2, \*temp;  p1 = &x;  p2 = &y;    \*p1 = \*p1 + \*p2;  \*p2 = \*p1 - \*p2;  \*p1 = \*p1 - \*p2;  printf("x = %d, y = %d\n", x, y);    temp = p1;  p1 = p2;  p2 = temp;  (\*p2)++ ;    printf("x = %d, y = %d\n", x, y);  return 0;  } | **x = 0, y = 25**  **x = 1, y = 25** |

|  |  |
| --- | --- |
| #include <stdio.h> // **P5: 9 points**  void first(double\*, double\*, double\*);  void second(double\*, double\*, double\*);  double third(double\*, double\*);  int main(){  double a[3]= {1, 2, 7};  first(a, &a[1], &a[2]);  for(int i=0; i<3; i++)  printf("%.2f\n", a[i]);  return 0; }  void first(double \*x, double \*y, double \*z){  \*x = \*x + \*y;  printf("%.2f\n",\*y);  second(x, y, z);  printf("%.2f\n",\*x);  \*y = \*x + \*y; }  void second(double \*n, double \*m, double \*L){  double \*t;  t = n; n = m; m = t;  \*L = third(n, t); }  double third(double \*j, double \*k){  printf("%.2f\n",\*k);  return \*j + (\*k)++; } | **2.00**  **3.00**  **4.00**  **4.00**  **6.00**  **5.00** |

|  |  |
| --- | --- |
| #include <stdio.h> // **P6: 6 points**  void check1(int a,int \*b, int i){  if(i%3==0)  a=\*b+1;  else  a = \*b;  }  void check2(int \*a,int \*b, int i){  if(i%2==0)  \*a=\*b+1;  else  \*a=\*b-1;  }  int main(void){  int i,x[4]={2,3,7,8};  int y[4]={0};  for (i = 0 ;i<2;i++ )  check1(y[i],&x[i],i);  for (i = 2 ;i<4;i++ )  check2(&y[i],&x[i],i);  for (i = 0 ;i<4;i++ )  printf("%d\n",y[i]);  return 0;  } | **0**  **0**  **8**  **7** |

**Question 2: (16 points )**

Write a C program to open an input file named ***myinput.txt***. The program counts the number of **short**, **medium** and **long** words in the file and the results are printed in an output file named ***myoutput.txt***. A word is classified by its length as follows:

**short word:** length <= 3 letters**; medium word:** length >= 4 and <= 8 letters**; long word:** length > 8 letters. Your program should handle file not found error. Minimize the number of statements used in your program. Note: Assume that the input file ***myinput.txt contains only words of letters separated by spaces or newline****.* **Below are samples of input and output files.**

|  |  |
| --- | --- |
| ***myinput.txt*** file | ***myoutput.txt*** file |
|  |  |

Your answer:

|  |
| --- |
| **#include <stdio.h>**  **int main(void) {**  **FILE \*infile, \*outfile;**  **char c;**  **int shortwords=0,mediumwords=0, longwords=0, length=0, status;**    **infile = fopen("myinput.txt","r");**  **outfile = fopen("myoutput.txt","w");**    **if (infile == NULL){**  **printf("Cannot open myinput.txt for reading \n");**  **exit(1);**  **}**  **status = fscanf(infile,"%c",&c);**  **while ( status != EOF ) {**  **if(c != ' ' && c != '\n')**  **length++;**  **else {**  **if(length <= 3) shortwords++;**  **else if(length <= 8) mediumwords++;**  **else longwords++;**  **length = 0;**  **}**  **status = fscanf(infile,"%c",&c);**  **}**  **fprintf(outfile,"Number of short words = %d\n", shortwords);**  **fprintf(outfile,"Number of medium words = %d\n", mediumwords);**  **fprintf(outfile,"Number of long words = %d\n", longwords);**  **fclose(infile);**  **fclose(outfile);**    **return 0;}** |

**Question 3: (7 pts)**

Write in C language, a function **func** thatreceives only 1 input argument n of positive integer. The function will return the integer value of the function based on the definition mentioned below. It does not print anything. *Hint*: Note that n % 3 = 1 for cases n = 1, 4, 7, …

**Note: Write the function implementation only. No main program.**

Your answer:

**int func(int n) {**

**if (n % 3 == 1)**

**return 27 - (n + 2)/3;**

**else if (n % 3 == 2)**

**return 4 + (n + 1)/3;**

**else if (n % 3 == 0)**

**return 7 + 2\*n/3;**

**}**

**Question 4: (18 points )**

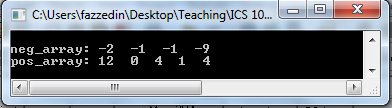
Write a function ***array\_split*** that takes the array ***original\_array*** and its ***size*** as input arguments and returns two arrays:

1. the array ***negative\_array*** that contains all negative integers in array ***original\_array***;
2. the array ***positive\_array*** that contains all non-negative integers in array ***original\_array***.

The function should also return the size of ***negative\_array*** and ***positive\_array.***

Complete the following main function so that the function ***array\_split*** is called and the results are printed as shown below.

Note: You are not allowed to use ***printf*** inside the function ***array\_split***



**#include <stdio.h>**

**#include <stdlib.h>**

**#define SIZE 9**

**int main() {**

**int org\_array[SIZE] = {12,-2,0,4,-1,1,-1,4,-9};**

**return 0;**

**}**

**//function implementation after this line**

Your answer:

**#include <stdio.h>**

**#include <stdlib.h>**

**#define SIZE 9**

**void split\_array(int x[SIZE], int neg[SIZE], int \*neg\_size, int pos[SIZE], int \*pos\_size);**

**int main() {**

**int org\_array[SIZE] = {12,-2,0,4,-1,1,-1,4,-9}, i,neg\_array\_size = 0, pos\_array\_size = 0;**

**int neg\_array[SIZE], pos\_array[SIZE];**

**split\_array(org\_array,neg\_array,&neg\_array\_size,pos\_array,&pos\_array\_size);**

**if (neg\_array\_size > 0 )**

**printf("neg\_array: ");**

**for (i=0; i< neg\_array\_size;i++)**

**printf("%d ",neg\_array[i]);**

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

**if (pos\_array\_size > 0 )**

**printf("pos\_array: ");**

**for (i=0; i< pos\_array\_size;i++)**

**printf("%d ",pos\_array[i]);**

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

**return 0;**

**}**

**//function definition after this line**

**void split\_array(int x[], int neg[], int \*neg\_size, int pos[], int \*pos\_size){**

**int i;**

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

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

**if (x[i] < 0) {**

**neg[\*neg\_size] = x[i];**

**\*neg\_size = \*neg\_size + 1;**

**}**

**else {**

**pos[\*pos\_size] = x[i];**

**\*pos\_size = \*pos\_size + 1;**

**}**

**}**

**Question 5: (17 points)**

Write a C program that *declares one integer array for storing student ids* called ***id\_array*** and a double array for storing student scores called ***scores\_array***. Your program starts by asking the user to:

1. enter the number of students,
2. read the student ids and their scores and stores them in the corresponding arrays,
3. compute and display the average, the highest mark along with the student id that scored the highest, and the lowest mark along with the student id that scored the lowest,
4. display the ids of students who are above average.

The results should be displayed correct to two decimal places. Assume that the maximum expected number of students is 100. The maximum number of students should be defined as a constant.

A sample run of your program must look like this:

Enter the number of students: 5

Enter the student ids and their scores:

200963670 93

200992650 97

200914730 90.5

200945730 76.5

200894600 80

Average = 87.40

Highest is 200992650 with a score of 97.00

Lowest is 200945730 with a score of 76.50

Students above average are:

200963670

200992650

200914730

Your answer:

**#include <stdio.h>**

**#include <stdlib.h>**

**#define SIZE 100**

**int main(void){**

**int id\_array[SIZE];**

**int n, i, max=0, min=0;**

**double scores\_array[SIZE];**

**double sum=0, avg;**

**printf("Enter the number of students: ");**

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

**printf("Enter the student ids and their scores:\n");**

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

**scanf("%d%lf", &id\_array[i], &scores\_array[i]);**

**sum += scores\_array[i];**

**if (scores\_array[i] > scores\_array[max]) max = i;**

**else if (scores\_array[i] < scores\_array[min]) min = i;**

**}**

**avg = sum / n;**

**printf("Average = %.2f\n", sum/n);**

**printf("Highest is %d with a score of %.2f\n", id\_array[max], scores\_array[max]);**

**printf("Lowest is %d with a score of %.2f\n", id\_array[min], scores\_array[min]);**

**printf("Students above average are:\n");**

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

**if (scores\_array[i] > avg)**

**printf("%d\n",id\_array[i]);**

**system("pause");**

**return 0;**

**}**