

CS 2073 Computer Programming with Engineering Applications

Fall 2008 – Final — May 9, 2008 1:30pm - 4:00 pm

Name:.....

Score:/30

Sequence number:.....

This exam has **six** questions. You have 120 minutes. Good luck...

1. **(5 points)** Complete the following C program that first reads **a**, **b**, **c**, where **a** and **b** are two degrees and **c** is increment. The program then generates a table of conversions from degrees to radians. Start degrees at **a** and increment by **c** until the degree is greater than **b**. Recall that we say 180° is equal to π radian. You can assume that π is 3.14.

```
#include <stdio.h>
#define PI 3.14
int main(void)
{
    /* Declare variables. If needed, you can declare more*/
    int a,b,c, degree;

    /* Enter the start degree, end degree, and increment */
    printf("Enter a b c : ");
    scanf("%d %d %d", &a, &b, &c);

    system("pause");

    /* Exit program. */
    return 0;
}
```

Name:.....

2. **(5 points)** Suppose a data file (say student.txt) contains student ID followed by nine numbers indicating the grades she/he got for each hw. As we will do in this class, we want to drop the lowest hw and compute the average for each student based on the best eight hw. For example, here is a sample file containing three students:

1	30	30	10	30	30	30	30	30	30
2	40	40	30	40	40	40	40	40	40
3	50	50	40	50	50	50	50	50	50

Write a program that can read the above file and output the student ID, and the average grade based on the best eight hw after dropping the lowest one.

(Hint: in a loop find the total of nine while finding the minimum of nine for each student, then you can take $(\text{total}-\text{min})/8$ as the average).

For the above file, your program should generate the following output on the screen:

```
Student Id      Average
-----
1              30
2              40
3              50
-----
```

Complete the code in the next page.

Name:

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    FILE *infp; /* declare other variables that you need */
```

```
    if((infp = fopen("student.txt", "r")) == NULL) exit(-1);
```

```
    fclose(infp);
```

```
    return 0;
```

```
}
```

Name:.....

3. (5 points) In this question you will trace the following program (i.e., run it by hand) and show how values in variables and arrays change in memory.

```
main()
{
  int x[3]={5, 2, 3};
  int y[3] = {0};
  int j;

  F1(x, y, 3);

  for(j = 0; j < 3; j++){
    printf("%d ", x[j]);
  }
  return 0;
}

void F1(int a[], int b[], int n)
{
  int i, j;

  for(i=0; i < n; i++){
    b[i] = F2(a, i);
  }
  return;
}

int F2(int a[], int i)
{
  int j, sum=0;

  for(j = 0; j < i+1; j++){
    sum = sum + a[j];
  }
  return sum
}
```

x[0]	5
x[1]	2
x[2]	3
y[0]	0
y[1]	0
y[2]	0
j	
a	
b	
n	
i	
j	
a	
i	
j	
sum	

OUTPUT:

Name:.....

4. (5 points) Suppose we declare `int a[N], b[M], c[N]`, where **N** and **M** are predefined global constants, somewhat we initialize **a** and **b** such that they contain two sets and assume that the values in **a** and **b** are already sorted in ascending order (so you don't need to anything else, but use them as is). NOW, We are interested in finding:
- the difference between **a** and **b**, and put the elements of **a-b** into **c** and
 - the number of elements in **c**.

Write a function that will take **a**, **b**, **c** as parameters and determine the difference set **c** and the number of elements in **c**. The function will return the number of elements in **c** while storing the difference between **a** and **b** into **c**.

For example, if **a** = {2, 5, 7, 8, 13} and **b** = {1, 5, 7, 9, 12}, then your function will return 3 as the number of elements in **c**, and we will have **c** = {2, 8, 13} as **a-b**.

```
int set_difference(int a[N], int b[M], int c[N])  
{
```

```
    return .....
```

```
}
```

Name:.....

5. **(5 points)** Write a function that takes two matrices as parameters (i.e., 2D arrays, say $A[Ra][Ca]$, and $B[Rb][Cb]$, where Ra , Ca , Rb , and Cb are predefined global constants). Then your function should check if these two matrices are equal (same) or not. If so, it should return 1; otherwise, it should return 0.

Two matrices are said to be equal if and only if they have the same order (same number of rows and same number of columns) and if the corresponding elements in both matrices are equal. For example,

$$A =$$

4	2	5
5	1	3

$$B =$$

4	2	5
5	1	3

are equal

```
int check_matix_equality( int A[Ra][Ca], int B[Rb][Cb])
{
```

```
}
```

Name:.....

6. (5points) Trace the following program. Using the below table **show how variables change** in the and **give the output**.

```
main()
{
    int x, y, z, *p1, *p2;

    p1 = &x;
    p2 = p1;
    x=4;
    y=5;

    z = f1(x, *p1, p2, &y);

    printf("x=%d   y=%d   z=%d "
           " and *p1=%d *p2=%d ",
           x, y, z, *p1, *p2);
}

int f1(int a, int b, int *c, int *d)
{
    int x, y;

    x = *c + a;
    y = *d - b;
    *c = x;
    *d = y;
    return x-y;
}
```

MEMORY

name	Address	Content/Value
x	11	
y	12	
z	13	
p1	14	
p2	15	
	16	
	...	
	58	
a	59	
b	60	
c	61	
d	61	
x	63	
y	64	
	65	

OUTPUT: