Indian Statistical Institute, Bangalore

CS1 – Final Exam, 2014-15

Total Marks: 100

1. **(3x5=15)** Given the following declaration:

```
struct student {
   int id;
   struct student * next;
};
struct student a,b,c;
struct student * p, * q, * head, * new;
```

Write the **C** statements / code for each of the following, assuming **head** is pointing to the first item of a linked list of items:

   a. Print the address and **id** field's value of the structure pointed to by **head**, and the address of the following structure.
   b. Print the **id** of all elements in the linked list starting from the item pointed to by **head**.
   c. Assuming that **new** is pointing to a new initialized structure; and assuming **p** is pointing to some structure in the linked list, insert the structure that new is pointing to into the linked list at the position *after* **p**. Use only two assignment statements.
   d. Assuming **p** is pointing to some structure in the linked list, delete the structure in the linked list at the position *after* **p**. Use only two assignment statements and one `free`.
   e. Interchange the **id** of the structure pointed to by **p** with that pointed to by **q.**

2. **(2x5=10)** Give declarations for the *variable* **x** in each of the following cases:
   a. **x** is a pointer to a variable of the type of **head** in 1. above.
   b. **x** is a an array of pointers to the structure defined in 1. above.
   c. **x** is a two-dimensional array of structures defined in 1. above, with 20 rows and 30 columns.
   d. **x** is a structure. It contains two fields named **a** and **b,** each of which are pointers to the structure defined in 1. above.
   e. **x** is a structure. It contains two fields named **a** and **b,** each of which are structures as defined in 1. above.

3. **(10)** You wish to sort an array of strings in lexicographically increasing order using insertion sort. Each element of the array is a pointer to a string. Here is the function prototype:

```
void insertSort(char * a[], int n);
```

Implement the above function. To compare strings use the library function **strcmp(char *a, char *b)**. The function **strcmp** returns a negative number, zero or a positive number to indicate that the string pointed to by **a** is lexicographically before, the same as, or after the string pointed to by **b.**

4. **(10)** Considering the following definitions: **`int a=1, b=2;  float c=5.8, d=2.0;`** answer the following questions:
    a. printf("%d",(a/b));
    b. printf("%f",(float)(a/b));
    c. printf("%f",(c/d));
    d. printf("%d",(int)(c/d));
    e. printf("%f",c/b);
    f. printf("%d",a==b);
    g. printf("%d",a=b);
    h. printf("%d",++a);
    i. printf("%d",a--);
    j. printf("%d",--a+b--);
5. **(10)** You wish to compute the greatest common divisor *gcd(a,b)* of two integers *a* and *b* *(a>=b)*. To do this use the following method:

    Compute *r* as the remainder of *a* divided by *b*.

    If *r* is zero then the *gcd(a,b)* is *b*.

    Else recursively compute *gcd(a,b)* as the *gcd(b,r)* .

    Eg. *gcd* (32,12) is computed as *gcd* (12,8) is computed as *gcd* (8,4) which is computed as 4, which is the answer.

    Write the *recursive* C function **`int gcd(int a, int b);`** which takes two integer parameters and returns the value of their *gcd*.

6. **(10)**Assume the following variable definitions:

```
int *p=malloc(500*sizeof(int));  char *q=malloc(500*sizeof(char));
int a[100]; char b[100];
struct { int x; int y; } z;
```

    If the sizes of integers, pointers and characters are represented by the values *i*, *p* and *c* respectively, express the value of the following expressions in terms of *i*, *p* and *c*:

    a) sizeof(q) b) sizeof(*q)
    c) sizeof(p) d) sizeof(*p)
    e) sizeof(a) f) sizeof(b)
    g) sizeof(z) h) sizeof(z.x)
    i) sizeof(z.y) j) sizeof(p+100)

7. **(15) D** is an **n**x**n** Matrix of distances of **n** points in a city. **`D[i][j]=`**the distance of travelling by road from point **`i`** to point **`j`**. For each point one can define **`M(i)`** as $Max_{\{j=0..n-1\}}$ **`D[i][j].`** Given **D,** the point **c** which has minimum value of M, i.e., **`M(c)<=M(i)`** over all **`i`** in {0 .. *n*-1} is called the center of D. Write the function:: **`int findCenter(int D[][20], int n);`** which returns such a center point.
8. **(10)** A student is given this problem: He will be given a sequence of about **`10,000`** integers where each is in the range **`0..99.`** So obviously there will be some repeated integers. His job is

to find out which integers do not exist. Write a program to do this by reading in the integers one by one (not storing them all) and simply keeping count of the number of times each integer in **`0..99`** is given.

9. **(3+3+4=10)** You wish to implement a way of doing complex number arithmetic.
    a. Design a way to organize data to represent a complex number, using arrays or structures. (Remember, real and imaginary parts are real numbers).
    b. Implement a function to return the absolute value (**`|a|`**) of the complex number **`a`**.
    c. Implement a function to compute and return e^{a+ib} which itself is a complex number. Use the fact that e^{ib}=sin(b)+*i*cos(b). Use the C library functions: **`exp(x), sin(x)`** and **`cos(x)`** that evaluate the corresponding mathematical functions e^x, *sin*(x) and *cos*(x). Your function allocates the required data to hold the computed complex number and returns a pointer to it.