Indian Institute of Technology, Kharagpur

Department of Computer Science and Engineering

Class Test 2, Autumn 2013-14

Programming and Data Structures (CS 11001)

Full marks: 50 **Date**: 04-Nov-13 **Time**: 1 hour

Name	Roll No.	Section	Marks Obtained

1. Answer ALL questions.
2. Answer all questions in the space provided in this question paper itself. Use the designated spaces for rough work.
3. Marks for every question is shown with the question.

---

1. Write the values of the `int` variables when the following program ends. **[6 * 2 = 12 marks]**

```
#include <stdio.h>

void main() {
    int a[] = {0, 1, 4, 9, 16};
    int *p, *q;
    int i, j, k, l, m, n;

    q = &a[4];
    p = q - 4;

    i = *q;
    j = *p++;
    k = *--q;
    l = p[1];
    m = *(q-2);
    n = q - p;
}
```

```
Answer:

    i = __________

    j = __________

    k = __________

    l = __________

    m = __________

    n = __________
```

ROUGH WORK

Use this space for your rough work, if any. This part will not be evaluated.

Answer:

i = 16

j = 0

k = 9

l = 4

m = 1

n = 2

2. Consider the following function and the snapshot of its local variables in the stack frame (activation record) at `Stack Frame Snapshot`:

```
void main() {
    int i = 3, *p = 0, **r = 0;
    double d = 2.0, e = 0.0, *q = &d;

    p = __________;

    r = __________;

    **r = __________;

    // Stack Frame Snapshot

    printf("Line 1: %p %p %p\n", &i, &d, &e);
    printf("Line 2: %d %lf %p\n", *p, *q, *r);
    printf("Line 3: %p %p %p\n", p, q, r);
    printf("Line 4: %p %p %p\n", &p, &q, &r);
}
```

Stack Frame of `main()` at `Stack Frame Snapshot`

Memory Address	Stack	Stored Variable
0x007AF960	0x007AF96C	p
0x007AF964	0x007AF960	r
0x007AF968	0x007AF970	q
0x007AF96C	4	i
0x007AF970	2.000000	d
0x007AF978	0.000000	e

(a) Fill up the blank lines in the program. **[1 + 1 + 2 = 4 marks]**

(b) Write the output of the function in the blank lines below. **[12 * 1 = 12 marks]**

```
Line 1: ____________________  ____________________  ____________________

Line 2: ____________________  ____________________  ____________________

Line 3: ____________________  ____________________  ____________________

Line 4: ____________________  ____________________  ____________________
```

ROUGH WORK

Use this space for your rough work, if any. This part will not be evaluated.

Answer:

```
p = &i;

r = &p;

**r = 4;
```

Answer:

```
Line 1: 0x007AF96C 0x007AF970 0x007AF978
Line 2: 4 2.000000 0x007AF96C
Line 3: 0x007AF96C 0x007AF970 0x007AF960
Line 4: 0x007AF960 0x007AF968 0x007AF964
```

Roll No:

3. Fill up the missing codes in the `char *DuplicateString(const char *str)` function that takes a C string, duplicates it and returns its pointer. **[1 + 1 + 1 + 2 + 2 + 1 = 8 marks]**

```
#include <stdio.h>

#include <_____________________>

char *DuplicateString(const char *str) {
    int n = 0;
    char *s = 0;

    // Compute the length of the string

    for(; __________; ++n);

    // Allocate space for duplicate string

    s = _____________ malloc(_______________________________);

    // Copy string

    for(n = 0; _____________________; ++n);

    // Return handle to duplicate string

    return _____;
}
```

ROUGH WORK

Use this space for your rough work, if any. This part will not be evaluated.

Answer:

```
#include <stdio.h>
#include <stdlib.h> or <malloc.h>
          --------      --------

char *DuplicateString(const char *str) {
    int n = 0;
    char *s = 0;

    // Compute the length of the string
    for(; str[n]; ++n);
          ------

    // Allocate space for duplicate string
    s = (char *)malloc(sizeof(char)*(n+1));
        --------       ------------------

    // Copy string
    for(n = 0; s[n] = str[n]; ++n);
               -------------

    // Return handle to duplicate string
    return s;
           --
}
```

4. Consider the following program.

```
void main() {
    char word[20], **head;
    int i, n;

    scanf("%d",&n);

    head = (char **)malloc
        (n * sizeof(char *));
    for (i=0; i<n; ++i) {
        scanf("%s", word);
        head[i] = (char *)malloc
            ((strlen(word)+1)*sizeof(char));
        strcpy(head[i], word) ;
    }

    // Process strings ...
}
```

ROUGH WORK

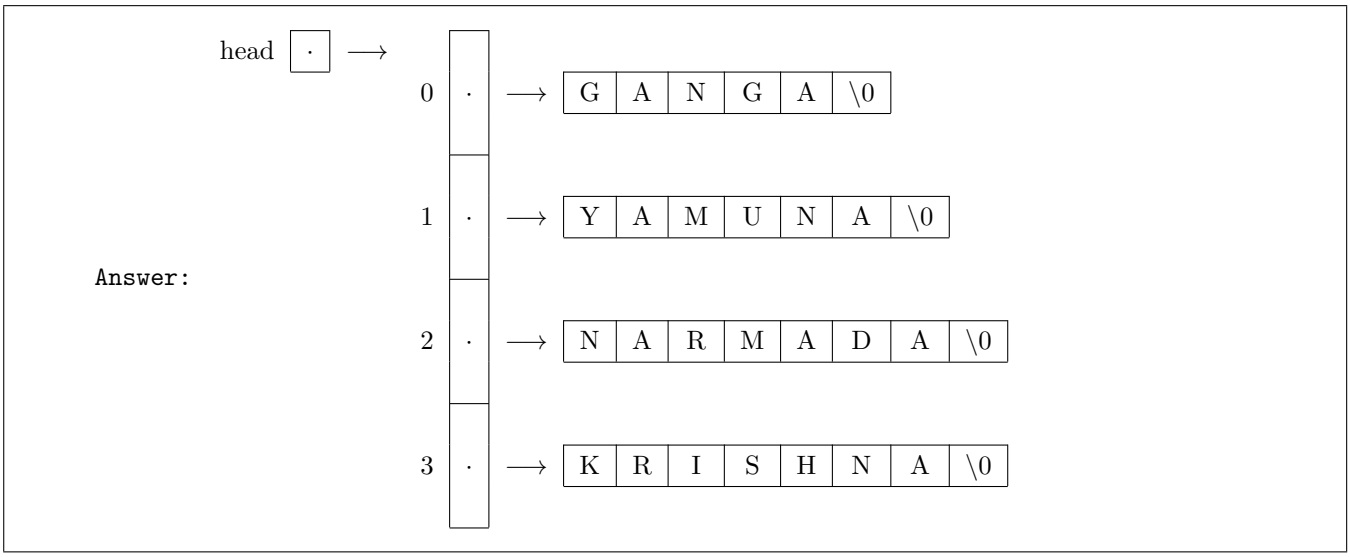
Use this space for your rough work, if any. This part will not be evaluated.

The following input is given to the program:

```
4
GANGA
YAMUNA
NARMADA
KRISHNA
```

Draw the dynamically created pointer configuration pointed to by `head` at `Process strings ...` execution point. [**5 marks**]

Answer:



Roll No:

5. In the context of a 2D point structure `typedef struct Point_ { double x; double y; } Point;` the function `Point CenterOfGravity(Point *pts, unsigned int n)` finds the center of gravity of an array `pts` of `n` `Point`s.

   (a) Fill up the missing codes below: **[6 * 1 = 6 marks]**

```
#include <stdio.h>

// Type to represent a 2D point
typedef struct Point_ { double x; double y; } Point;

// Function to print a point
void Print(Point pt) {
    printf("\t(%lf, %lf)\n", pt.x, pt.y);
}

// Function to compute CG
Point CenterOfGravity(Point *pts, unsigned int n) {
    unsigned int i;
    Point cg = ________________; // Intialize cg

    // Print the input points
    printf("Input points:\n");
    for(i = 0; i < n; ++i) Print(pts[i]);

    // Compute the sum of x and y coordinates respectively
    for(i = 0; i < n; ++i, ++pts) {

        cg.x += ________________;

        cg.y += ________________;
    }

    // Normalize the sum by the number of points to find CG

    cg.x /= ________________;

    cg.y /= ________________;

    return cg;
}

void main() {
    Point points[] = {2, 5, 7, 5, 3, 6, 4, 4};

    // Compute CG for points[]
    Point cg = CenterOfGravity(points, ________________________________________);

    printf("Center of Gravity:\n");
    Print(cg);
}
```

(b) Write the output of the program: **[0.5*4 + 1 = 3 mark]**

```
Input points:

        (__________, __________)

        (__________, __________)

        (__________, __________)

        (__________, __________)

Center of Gravity:

        (__________, __________)
```

ROUGH WORK

Use this space for your rough work, if any. This part will not be evaluated.

Answer:

```
#include <stdio.h>

// Type to represent a 2D point
typedef struct Point_ { double x; double y; } Point;

// Function to print a point
void Print(Point pt) {
    printf("\t(%lf, %lf)\n", pt.x, pt.y);
}

// Function to compute CG
Point CenterOfGravity(Point *pts, unsigned int n) {
    unsigned int i;
    Point cg = {0, 0}; // Intialize cg
               ------

    // Print the input points
    printf("Input points:\n");
    for(i = 0; i < n; ++i) Print(pts[i]);

    // Compute the sum of x and y coordinates respectively
    for(i = 0; i < n; ++i, ++pts) {
        cg.x += pts->x;
                ------
        cg.y += pts->y;
                ------
    }

    // Normalize the sum by the number of points to find CG
    cg.x /= n;
            --
    cg.y /= n;
            --

    return cg;
}

void main() {
    Point points[] = {2, 5, 7, 5, 3, 6, 4, 4};

    // Compute CG for points[]
    Point cg = CenterOfGravity(points, sizeof(points)/sizeof(Point));
                                       ----------------------------

    printf("Center of Gravity:\n");
    Print(cg);
}
```

```
Answer:

Input points:
        (2.000000, 5.000000)
        (7.000000, 5.000000)
        (3.000000, 6.000000)
        (4.000000, 4.000000)
Center of Gravity:
        (4.000000, 5.000000)
```