End Semester Examination
Introduction to Computing (CS 101)
IIT Guwahati
April 28, 2015
Duration: 3 Hours
Question Paper Code: A

Instructions:

(1) Encircle the answer (A/B/C/D/E) in the space provided in the answer script.
(2) Each question has exactly one correct answer.
(3) There are a total of 50 questions. **Each question carries two marks.**
(4) **Negative Marking:** Each wrong answer will be penalized by $\frac{1}{2}$ **negative marks**.
(5) **Unless specified otherwise, assume that the size of a pointer, an integer, or a float equals to 4 bytes; size of a character is 1 byte; size of a short is 2 bytes; and the size of a double is 8 bytes.**
(6) In any question, 'all of the above' or 'none of the above' takes precedence over all other correct choices.
(7) Only portable ANSI C constructs are permitted.
(8) Assume standard RAM model of computation.
(9) Please write your name and roll number on both the question paper and the answer script.
(10) Please write your question paper code on the answer script.
(11) Possession of either a mobile phone or a calculator is strictly prohibited during the exam.
(12) No doubts will be entertained during the exam.
(13) There are 7 pages of questions and 1 page for rough work in the question paper. Please contact the invigilator, if otherwise.

Name: **Roll No:**

---

1. Identify the equivalent postfix representation for the following infix expression: $A * B * C + D * E * F$

   (A) $ABC * * + DEF * *$
   (B) $ABC * DEF * * * +$
   (C) $ABCD + * * EF * *$
   (D) $ABC * + D * EF * *$
   (E) None of the above

2. Evaluate the postfix expression: 2 3 * 4 5 * + 6 *

   (A) 126
   (B) 156
   (C) 420
   (D) 720
   (E) None of the above

3. Suppose we have some distinct numbers between 1 and 1000 stored in a sorted array, and we want to search the number 200 using binary search. Which of the following is a valid binary search sequence?

   (A) $100, 400, 300, 600, 200$
   (B) $400, 800, 700, 600, 200$
   (C) $200, 100, 600, 800, 200$
   (D) $300, 900, 700, 800, 200$
   (E) None of the above

4. Assume A to be an array of integers (e.g., "int a[5] = {1, 2, 3, 4, 5};"). Which of the following statements is invalid?

   (A) a[1]=3;
   (B) a++;
   (C) int *b=a;
   (D) int *b=a+1;
   (E) None of the above

5. What does the following code segment print?

```
#include<stdio.h>

int main()
{
 int a[4][2] = {1,2,3,4,5,6,7,8};
 int *ptr[4] = {a[0],a[1],a[2],a[3]};
 int i;

 for(i=0; i<4; ++i)
        printf("%d,", *ptr[i]);

 return 0;
}
```

   (A) $1, 2, 3, 4,$
   (B) $1, 3, 5, 7,$
   (C) $1, 1, 1, 1,$
   (D) Garbage values
   (E) None of the above

6. What does the following code segment print?

```
#include<stdio.h>

int main()
{
 int a[] = {10, 20, 30, 40, 50};
 int *p=a, *q=a;
 int i;

 for(i=0; i<4; ++i)
        printf("%d,%d,", *p++, *++q);

 printf("\n");
 return 0;
}
```

(A) $10, 20, 20, 30, 30, 40, 40, 50,$

(B) $10, 10, 20, 20, 30, 30, 40, 40,$

(C) $20, 10, 30, 20, 40, 30, 50, 40,$

(D) Garbage values

(E) None of the above

7. What does the following code segment print? Note that the ASCII values of a and b are 97 and 98, respectively.

```
#include<stdio.h>

typedef struct {

        int i;
        char c;
        } values1;

typedef union {

        int i;
        char c;
        } values2;


int main()
{
 values1 var1={0, 'a'};
 values2 var2={1, 'b'};
 printf("%d,%d,", var1.i, var1.c);
 printf("%d,%d", var2.i, var2.c);
 return 0;
}
```

(A) $0, a, 1, b$

(B) $0, 97, 1, b$

(C) $0, a, 1, 98$

(D) $0, 97, 1, 98$

(E) None of the above

8. The output of the following code segment is $0xbf95288c$,_____,_____,_____. Choose the correct option for filling up the blanks.

```
#include<stdio.h>
void main()
{
 int a[5] = {10, 20, 30, 40, 50};
 printf("%p,%p,", a, &a);
 printf("%p,%p", a+1, &a+1);
}
```

(A) $0xab687a80, 0xbf952890, 0xbf9528a0$

(B) $0xab687a80, 0xbf952890, 0xab687a90$

(C) Compile time error

(D) Run time error

(E) None of the above

9. Consider an empty stack S in which the following sequence of operations are performed: Push(1), Pop(), Push(3), Push(4), Pop(), Push(2), Pop(), Push(9), Push(5), Push(8), Pop().

   Which of the elements are present in the stack (in any order) after the above operations?

(A) $3, 9, 5$

(B) $9, 5, 8$

(C) $1, 3, 4$

(D) $3, 2, 9$

(E) None of the above

10. Consider an empty queue Q in which the following sequence of operations are performed: Enqueue(7), Enqueue(6), Dequeue(), Enqueue(3), Enqueue(9), Enqueue(1), Enqueue(2), Dequeue(), Enqueue(4).

    Which of the elements are present in the queue (in any order) after the above operations?

(A) $7, 6, 3, 9, 4$

(B) $7, 3, 9, 1, 4$

(C) $3, 9, 1, 2, 4$

(D) $6, 3, 9, 1, 4$

(E) None of the above

11. Consider a singly linked list L in which every node (structure) consists of three members. The first member is an integer data and the second member is a character. Identify the correct data type for the third member.

(A) Integer

(B) Pointer to an integer

(C) Pointer to a character

(D) Pointer to a node

(E) None of the above

12. What does the following code segment print?

```
#include <stdio.h>
void swap1(int a, int b)
{
        int temp;
        temp = a;
        a = ++b;
        b = temp;
}
void swap2(int *a, int *b)
{
        int temp;
        temp = *a;
        *a = ++*b;
        *b = temp;
}
void main()
{
 int i=10, j=20, k=30, l=40;
 swap1(i, j);
 swap2(&k, &l);
 printf("%d,%d,%d,%d\n",i,j,k,l);
}
```

(A) $21, 10, 30, 40$

(B) $21, 10, 30, 40$

(C) $10, 20, 41, 30$

(D) $20, 10, 40, 30$

(E) None of the above

13. Identify the number of bytes required to store the character 'c' and the string "c", respectively, in a C program.

(A) $1, 1$

(B) $1, 2$

(C) $2, 1$

(D) $2, 2$

(E) None of the above

14. Given an integer array declared by "int $A[3][5]$;", which of the following statements **does not** represent the element $A[2][3]$?

(A) $(*(A[2] + 3))$

(B) $(*(A + 2))[3]$

(C) $(*((*(A + 2)) + 3))$

(D) $(A[2] + 3)$

(E) None of the above

15. Given an integer array declared by

"int $A[\ ] = \{2, 4, 4, 8, 3\}$;", what is the value of $A[A[A[0]]]$?

(A) 2

(B) 4

(C) 6

(D) 8

(E) None of the above

16. Identify the 2's complement representation of the negative integer -5, assuming that an integer takes 8 bits.

(A) 10000001

(B) 01111011

(C) 11111011

(D) 11111010

(E) None of the above

17. The number of 1's in the binary representation of the integer $13 * 256^3 + 11 * 64^2 + 9 * 16 + 3$ is:

(A) 7

(B) 8

(C) 9

(D) 10

(E) None of the above

18. Given an integer array declared by "int $A[10]$;", what is the value of $((\&A[0]) - (\&A[9]))$? Assume that an integer takes 4 bytes.

(A) -9

(B) -10

(C) -36

(D) -40

(E) None of the above

19. If two pointers $p1$ and $p2$ are declared by "float *p1=NULL, *p2=NULL;", identify the invalid statement.

(A) $p1 = p1 + 2;$

(B) $p1 = p1 - 2;$

(C) $p1 = p1 + p2;$

(D) $int\ i = p1 - p2;$

(E) None of the above

20. Which of the following operations **can not** be implemented using constant auxiliary space (i.e., excluding the space occupied by the input) for a singly linked list whose head and tail pointers (respective pointers to the first and last node in the list) are given?

(A) Insert an item at the front of the list

(B) Insert an item at the rear of the list

(C) Delete the front item from the list

(D) Delete the rear item from the list

(E) None of the above

21. What is the expression for denoting the address of the array element $A[i][j]$ in an integer array $A[10][10]$?

(A) A+i+j

(B) *(A+i)+j

(C) *((A+i)+j)

(D) *(A+i+j)

(E) None of the above

22. How many '*' will be printed by the following program?

```
#include<stdio.h>
 void quiz(int i)
    {
        if (i > 1)
        {
            quiz(i % 2);
            quiz(i / 2);
            quiz(i % 2);
        }
        printf("*");
    }

void main()
{
        quiz(7);
}
```

(A) 3
(B) 5
(C) 7
(D) 9
(E) None of the above

23. How many '*' will be printed by the following program?

```
#include<stdio.h>
 void quiz(int i)
    {
        if (i > 1)
        {
            quiz(i / 2);
            quiz(i % 2);
            quiz(i / 2);
        }
        printf("*");
    }

void main()
{
        quiz(7);
}
```

(A) 3
(B) 5
(C) 7
(D) 9
(E) None of the above

24. Identify the declaration of a *four dimensional array* that does not give any compile-time error.

(A) int A[ ][2][2][2]={2,2,2,2};
(B) int A[ ][ ][ ][2]={2,2,2,2};
(C) int A[2][2][2][ ]={2,2,2,2,2,2,2,2};
(D) int A[2][ ][ ][ ]={2,2,2,2};
(E) None of the above

25. Identify the invalid statement that assigns a value to member1 of var object in the following code segment.

```
#include<stdio.h>
struct test{
        char member1;
} var;
void main()
{
 struct test *ptr=&var;
 /*identify the invalid statement.*/
}
```

(A) (*ptr).member1=97;
(B) *ptr.member1='a';
(C) (ptr)->member1='a';
(D) ptr->member1='a';
(E) None of the above

26. What does the following code segment print?
int a[2][3][4];
printf("%d, %d", sizeof(a[1]), sizeof(a[1][2]));

(A) 48, 16
(B) 24, 16
(C) 12, 16
(D) compile time error
(E) run-time error

27. Given that the address of a[0][0][0] is 0xbffb74d0, what does the following code segment print?
double a[2][3][4];
printf("%p, %p", &a[1][1][1], &a[1][2][0]);

(A) 0xbffb7500, 0xbffb7530
(B) 0xbffb7508, 0xbffb7560
(C) 0xbffb7558, 0xbffb7570
(D) 0xbffb74d8, 0xbffb7560
(E) 0xbffb74f8, 0xbffb7578

28. Given that the address of z[1] is 0xbf94add0, what does the following code segment print?

int z[3][4];
printf("%p, %p", z, z+1);

(A) 0xbf94adb0, 0xbf94adf0
(B) 0xbf94add0, 0xbf94add0
(C) 0xbf94add0, 0xbf94adf0
(D) 0xbf94adc0, 0xbf94add0
(E) 0xbf94adc0, 0xbf94adf0

29. Find the appropriate ones to replace X and Y in the following code, respectively:

double (*p)[2];
p = (X) calloc(5, sizeof(Y));

(A) double [2], double (*)[2]
(B) double (*)[2], double (*)[2]
(C) double (*)[2], double[2]
(D) void *, void *
(E) void *, double (*)[2]

30. In the following code, which ones are the appropriate replacements for X, Y, and Z, respectively?
double *buf[2];
buf[0] = (X) malloc(20*sizeof(double));
double d[2] = {11, 12};
buf[1] = (Y) &d[0];
free((Z) buf[0]);

    (A) void*, double*, double*

    (B) double*, void*, void

    (C) double*, typecast (Y) not required, double

    (D) void*, double, typecast (Z) not required

    (E) double*, typecast (Y) not required, typecast (Z) not required

31. Given that just before calling func in the main function, 0x951b008 is stored in head, what does the following code output?

```
#include<stdio.h>
#include<stdlib.h>
typedef struct {
   int capacity;
   int topelem;
   void **ptr;
} Stack;
void func(Stack **head) {
  head += 4;
  printf("%p,", head);
}
int main(void) {
   Stack **head;
   head = (Stack**)
     malloc(5*sizeof(Stack*));
   func(head);
   head++;
   printf("%p\n", head);
}
```

    (A) 0x951b00c,0x951b01c

    (B) 0x951b00c,0x951b00d

    (C) 0x951b018,0x951b01c

    (D) 0x951b00c,0x951b009

    (E) 0x951b018,0x951b00c

32. Let a function *func* accept 'double [3][4]' as its first argument. Then the valid declaration of that formal argument is:

    (A) double b[][4]

    (B) double b[3][4]

    (C) double (*b)[4]

    (D) all of the above

    (E) (A) and (B) only

33. Consider the prototype of *strcpy* function: 'char *$strcpy$(char *p, const char *q)'. Let buf1 be the block of memory pointed by p. And, let buf2 be the block of memory pointed by q. In calling strcpy function, buf1 and buf2 are respectively allocated by

    (A) *strcpy* function, *strcpy* function

    (B) function that calls *strcpy*, *strcpy* function

    (C) *strcpy* function, function that calls *strcpy*

    (D) function that calls *strcpy*, function that calls *strcpy*

    (E) both must be allocated using either *malloc* or *calloc* by the function that calls *strcpy*

34. Consider the prototype of *strdup* function: 'char *$strdup$(const char *p)'. Let buf1 be the block of memory pointed by p. Let buf2 be the block of memory pointed by the pointer returned by strdup. The buf1 and buf2 are respectively

    (A) allocated by the function that calls *strdup*, allocated by *strdup*

    (B) allocated by *strdup*, allocated by the function that calls *strdup*

    (C) allocated by the function that calls *strdup*, allocated by the function that calls *strdup*

    (D) required to be allocated on stack

    (E) required to be allocated on heap

35. After returning from *strcat* function (whose prototype is 'char *strcat(char *dest, const char *src)'), the function that calls strcat function

    (A) needs to call $free$ function twice: to free the memory pointed by dest and to free the memory pointed by src.

    (B) needs to call $free$ function only once: to free the memory pointed by dest.

    (C) needs to call $free$ function only once: to free the memory pointed by src.

    (D) does not need to call $free$ function to free memory pointed by either of dest and src.

    (E) needs to free the memory pointed by src and dest depending on how those memory blocks are allocated.

36. Consider the prototype of a function: 'Node *deleteNode(Node *head, void *p, Helper func)'. Here, Node is the data type of any node of a doubly-linked list. Apart from *prev* and *next* pointers, each node also has a pointer to an object (of any type). Helper is a pointer to a function, wherein that function has two arguments: each is a pointer to an object of any type; and, that function returns pointer to an object of any type. The respective declarations of Helper type and the prototype of any function that has Helper type are:

    (A) typedef void *Helper(void*, void*),
    void* func(void*, void*)

    (B) typedef void (*Helper)(void),
    void func(void)

    (C) typedef void *(*Helper)(void*, void*),
    void* func(void*, void*)

    (D) typedef void (*Helper(void*, void*)),
    void func(void*, void*)

    (E) typedef void *(*Helper(void*, void*)),
    void func(void)

37. What does the following code segment print?
struct Point {double x; int y; void *data;};
printf("%d, %d, %d",
sizeof(struct Point),
sizeof(struct Point (*) [2]),
sizeof(struct Point * [2]));

(A) 16, 8, 8
(B) 16, 16, 8
(C) 16, 4, 8
(D) 8, 8, 16
(E) 8, 8, 8

38. Let *Part* be the name of a user-defined data type. Given that 'Part *parts[4]', which of the following must be in the memory explicitly allocated (for example, using malloc) in the program?

(A) *(parts+2)
(B) *parts
(C) parts[1]
(D) parts
(E) none of the above

39. Let *PartA* and *PartB* be two user-defined data types. Assume that the size of PartA is 16 bytes and the size of PartB is 32 bytes. Given that 'union {PartA a1; PartB b; PartA a2;} obj;' and the address of obj.a2 is 0x1234d0, the addresses of obj.a1 and obj.b are respectively:

(A) 0x1234b0, 0x1234d0
(B) 0x1234d0, 0x1234d0
(C) 0x1234b0, 0x1234a0
(D) 0x1234a0, 0x1234b0
(E) none of the above

40. The following is the prototype of fprintf:

(A) int fprintf(const FILE *, char *format, ...)
(B) int fprintf(FILE *, char *format, ...)
(C) int fprintf(const FILE *, const char *format, ...)
(D) int fprintf(FILE *, const char *format, ...)
(E) none of the above

41. For any large positive integer n, the respective tight asymptotic worst-case and best-case upper bounds on the time complexities in computing the sum of cubes of n numbers (each of type double) given in contiguous entries of an array are:

(A) $O(n), O(n)$
(B) $O(n), O(\lg n)$
(C) $O(n^3), O(n^3)$
(D) $O(n^3), O(n)$
(E) $O(n^3), O(1)$

42. Given n distinct integers in an array A and an integer k, identify the asymptotically tight worst-case time complexity of the most efficient algorithm that determines whether k is the sum of two integers from A:

(A) $O(n^3)$
(B) $O(n \lg n)$
(C) $O(n^2)$
(D) $O(1)$
(E) $O(\lg n)$

43. Among the following macros, which one is appropriate to square a double?

(A) #define SQUARE(x) x*x
(B) #define SQUARE(x) (x)*(x)
(C) #define SQUARE(x) x ^ 2
(D) #define SQUARE(x) (x) ^ (2)
(E) none of the above

44. Suppose the push and pop operations of a stack data structure are efficiently implemented using object(s) of a fixed-size circular queue (queue that uses array in circular fashion) data structure. Also, suppose that the push operation is asymptotically not costlier than the pop operation of that stack. Assuming that there are n pointers (of type void*) are placed on the stack (for any large positive integer n), the respective tight asymptotic upper bounds on time complexities of push and pop are:

(A) $O(n^2)$, $O(n)$
(B) $O(n)$, $O(n^2)$
(C) $O(n)$, $O(1)$
(D) $O(1)$, $O(1)$
(E) $O(1)$, $O(n)$

45. Suppose that the enqueue and dequeue operations of a queue data structure are efficiently implemented using object(s) of a fixed-size stack data structure. Also, suppose that the enqueue operation is asymptotically not costlier than the dequeue operation of that queue. Assuming that there are n pointers (of type void*) are placed on the queue (for n being any large positive integer), the respective tight asymptotic upper bounds on time complexities of enqueue and dequeue are:

(A) $O(n^2)$, $O(n)$
(B) $O(n)$, $O(n^2)$
(C) $O(n)$, $O(1)$
(D) $O(1)$, $O(1)$
(E) $O(1)$, $O(n)$

46. Identify two data structures whose implementation using the structure construct (of *C* programming language) requires to have a pointer to itself as a member (i.e., self-referential structure):

(A) stack, queue
(B) queue, circular queue
(C) singly-linked list, doubly-linked list
(D) array, pointer to circular queue
(E) array of pointers to void*, dynamic array of pointers to void*

47. Consider a function $func$ that accepts one argument, say h, which points to any node of a linked list and the $func$ is assumed to be implemented efficiently to remove a node before the one pointed by h. Assuming that there are n nodes (n being asymptotically large) in the linked list L pointed by h, the tight asymptotic upper bound on the time complexities when L is a singly-linked list and L is a doubly-linked list, respectively:

    (A) $O(1)$, $O(1)$
    (B) $O(n)$, $O(n)$
    (C) $O(1)$, $O(n)$
    (D) $O(n)$, $O(1)$
    (E) none of the above

48. Let L be a linked list. Let every node v of L point to the first character of some constant string; and assume that any such string has at most k characters. Consider an efficient algorithm A that takes head (pointer to the first node) of list L as its input and returns 1 whenever every string pointed by every node of the list L is a palindrome. Otherwise, A returns 0. Assuming that the list L has n nodes, the tight upper bound on the worst-case asymptotic time complexities when L is a singly-linked list and when L is a doubly-linked list are respectively:

    (A) $O(nk)$, $O(nk)$
    (B) $O(max(n,k))$, $O(min(n,k))$
    (C) $O(n+k)$, $O(n+k)$
    (D) $O(max(n,k))$, $O(nk)$
    (E) $O(nk)$, $O(min(n,k))$

49. Given n points with (x,y) coordinates in two-dimensional Euclidean plane, the tight worst-case asymptotic upper bound on the time and auxiliary space (i.e., excluding the space occupied by the input) complexities of the most efficient algorithm to output a pair of points that are closest among all the input points are respectively

    (A) $O(n^2)$, $O(n^2)$
    (B) $O(n^2)$, $O(1)$
    (C) $O(n^2)$, $O(n)$
    (D) $O(n)$, $O(n)$
    (E) $O(1)$, $O(1)$

50. Let Q be an object of a queue data structure, which uses array in circular fashion and the array is expandable (dynamic-sized circular queue). Suppose in enqueing n objects to Q, the capacity of Q got doubled (that is, whenever queue is full and the user wants to enqueue a new object) k times and the Q is full when all the n objects are enqueued. Assume that the number of dequeues are upper bounded by the number of enqueues; and assume that no dequeue causes reducing the size of the array. The respective tight worst-case asymptotic upper bounds on the time and space complexities when all the enqueue and dequeue operations are considered together is:

    (A) $O(n)$, $O(1)$
    (B) $O(n^2)$, $O(n)$
    (C) $O(n)$, $O(n)$
    (D) $O(n \lg n)$, $O(n \lg n)$
    (E) $O(n \lg n)$, $O(n)$