Bundelkhand Institute of Engineering and Technology, Jhansi

Computer Science Students' Council (CoSSco) 2k19-20

Technical Aptitude Questions

Max. Time: 60 mins

Instructions:

- There are total 30 questions.
- Maximum time allotted is 60 minutes.
- There is only one option correct for each questions.
- Each question carries of 3 marks.
- There will be 1/3rd negative marking in case of wrong answer.
- No marks will be deducted or provided in case of unattempted question.
- All questions should be done on the basis of C language.

1. A function f defined on stacks of integers satisfies the following properties. f(Ø) = 0 and f (push (S, i)) = max (f(S), 0) + i for all stacks S and integers i.
If a stack S contains the integers 2, -3, 2, -1, 2 in order from bottom to top, what is f(S)?

a) 6 b) 4
c) 3 d) 2

2. Consider the following C program:

```
#include<stdio.h>
#define EOF -1
void push (int);
int pop  (void);
void flagError ();
int main ()
{       int c, m, n, r;
        while ((c = getchar ()) != EOF)
        { if  (isdigit (c) )
                push (c);
         else if ((c == '+') || (c == '*'))
        {       m = pop ();
                n = pop ();
                r = (c == '+') ? n + m : n*m;
                push (r);
         }
         else if (c != ' ')
                flagError ();
  }
   printf("% c", pop ());
}
```

What is the output of the program for the following input ? 5 2 * 3 3 2 + * +

a) 15 b) 25
c) 30 d) 15

3. Let S be a stack of size n >= 1. Starting with the empty stack, suppose we push the first n natural numbers in sequence, and then perform n pop operations. Assume that Push and Pop operation take X seconds each, and Y seconds elapse between the end of one such stack operation and the start of the next operation. For m >= 1, define the stack-life of m as the time elapsed from the end of Push(m) to the start of the pop operation that removes m from S. The average stack-life of an element of this stack is

a) n(X+Y) b) 3Y+2X
c) n(X+Y)-X d) Y+2X

4. A single array A[1..MAXSIZE] is used to implement two stacks. The two stacks grow from opposite ends of the array. Variables top1 and top2 (topl< top 2) point to the location of the topmost element in each of the stacks. If the space is to be used efficiently, the condition for "stack full" is

a) (top1 = MAXSIZE/2) and (top2 = MAXSIZE/2+1)

b) top1 + top2 = MAXSIZE
c) (top1= MAXSIZE/2) or (top2 = MAXSIZE)
d) top1= top2 -1

5. Stack A has the entries a, b, c (with a on top). Stack B is empty. An entry popped out of stack A can be printed immediately or pushed to stack B. An entry popped out of the stack B can be only be printed. In this arrangement, which of the following permutations of a, b, c are not possible?

a) b a c
b) b c a
c) c a b
d) a b c

6. An implementation of a queue Q, using two stacks S1 and S2, is given below:

```
void insert(Q, x)
{
  push (S1, x);
}
 void delete(Q){
  if(stack-empty(S2)) then
    if(stack-empty(S1)) then
   {
      print("Q is empty");
      return;
    }
    else while (!(stack-empty(S1))){
      x=pop(S1);
      push(S2,x);
    }
  x=pop(S2);
}
```

Let n insert and m (<=n) delete operations be performed in an arbitrary order on an empty queue Q. Let x and y be the number of push and pop operations performed respectively in the process. Which one of the following is true for all m and n?

a) n+m <= x < 2n and 2m <= y <= n+m
b) n+m <= x < 2n and 2m<= y <= 2n
c) 2m <= x < 2n and 2m <= y <= n+m
d) 2m <= x <2n and 2m <= y <= 2n

7. Consider the following pseudo code. Assume that IntQueue is an integer queue. What does the function fun do?

```
void fun(int n)
{
  IntQueue q = new IntQueue();
  q.enqueue(0);
  q.enqueue(1);
  for (int i = 0; i < n; i++)
  {
    int a = q.dequeue();
    int b = q.dequeue();
    q.enqueue(b);
    q.enqueue(a + b);
    ptint(a);
  }
}
```

a) Prints numbers from 0 to n-1
b) Prints numbers from n-1 to 0
c) Prints first n Fibonacci numbers
d) Prints first n Fibonacci numbers in reverse order.

8. Let Q denote a queue containing sixteen numbers and S be an empty stack. Head(Q) returns the element at the head of the queue Q **without** removing it from Q. Similarly Top(S) returns the element at the top of S **without** removing it from S. Consider the algorithm given below.

```
while Q is not Empty do
    if S is Empty OR Top(S) ≤ Head(Q) then
        x := Dequeue(Q);
        Push(S,x);
    else
        x := Pop(S);
        Enqueue(Q,x);
    end
end
```

The maximum possible number of iterations of the while loop in the algorithm is_____

a) 16 b) 32
c) 256 d) 64

9. Suppose you are given an implementation of a queue of integers. The operations that can be performed on the queue are:

i. isEmpty (Q) — returns true if the queue is empty, false otherwise.
ii. delete (Q) — deletes the element at the front of the queue and returns its value.
iii. insert (Q, i) — inserts the integer i at the rear of the queue.

Consider the following function:

```
void f (queue Q)
{
int i ;
if (!isEmpty(Q))
{
  i = delete(Q);
  f(Q);
  insert(Q, i);
 }
}
```

What operation is performed by the above function f?

a) Leaves the queue Q unchanged
b) Reverses the order of the elements in the queue Q
c) Deletes the element at the front of the queue Q and inserts it at the rear keeping the other elements in the same order
d) Empties the queue Q

10. A 3-ary max heap is like a binary max heap, but instead of 2 children, nodes have 3 children. A 3-ary heap can be represented by an array as follows: The root is stored in the first location, a[0], nodes in the next level, from left to right, is stored from a[1] to a[3]. The nodes from the second level of the tree from left to right are stored from a[4] location onward. An item x can be inserted into a 3-ary heap containing n items by placing x in the location a[n] and pushing it up the tree to satisfy the heap property. Which one of the following is a valid sequence of elements in an array representing 3-ary max heap?

a) 1, 3, 5, 6, 8, 9 b) 9, 6, 3, 1, 8, 5
c) 9, 3, 6, 8, 5, 1 d) 9, 5, 6, 8, 3, 1

11. An operator delete(i) for a binary heap data structure is to be designed to delete the item in the i-th node. Assume that the heap is implemented in an array and i refers to the i-th index of the array. If the heap tree has depth d (number of edges on the path from the root to the farthest leaf), then what is the time complexity to re-fix the heap efficiently after the removal of the element?

a) O(1) b) O(d) but not O(1)
c) $O(2^d)$ but not O(d) d) $O(d2^d)$ but not $O(2^d)$

12. Consider a hash table of size seven, with starting index zero, and a hash function (3x + 4)mod7. Assuming the hash table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8, 10 is inserted into the table using closed hashing? Note that '_' denotes an empty location in the table.

a) 8, _, _, _, _, _, 10 b) 1, 8, 10, _, _, _, 3
c) 1, _, _, _, _, _,3 d) 1, 10, 8, _, _, _, 3

13. Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020?

a) h(i) =i^2 mod 10 b) h(i) =i^3 mod 10
c) h(i) = (11 ∗ i^2) mod 10 d) h(i) = (12 ∗ i) mod 10

14. Consider an undirected unweighted graph G. Let a breadth-first traversal of G be done starting from a node r. Let d(r, u) and d(r, v) be the lengths of the shortest paths from r to u and v respectively, in G. lf u is visited before v during the breadth-first traversal, which of the following statements is correct?

a) d(r, u) < d (r, v) b) d(r, u) > d(r, v)
c) d(r, u) <= d (r, v) d) None of the above

15. An array A consists of n integers in locations A[0], A[1] ....A[n-1]. It is required to shift the elements of the array cyclically to the left by k places, where 1 <= k <= (n-1). An incomplete algorithm for doing this in linear time, without using another array is given below. Complete the algorithm by filling in the blanks. Assume alt the variables are suitably declared.

```
min = n; i = 0;
while (_____________) {
   temp = A[i]; j = i;
   while (_________) {
   A[j] = _________
   j= (j + k) mod n ;
   If ( j< min ) then
      min = j;
}
A[(n + i — k) mod n] = _____________
i = _____________
```

a) i > min; j!= (n+i)mod n; A[j + k]; temp; i + 1 ;
b) i < min; j!= (n+i)mod n; A[j + k]; temp; i + 1;
c) i > min; j!= (n+i+k)mod n; A[(j + k)]; temp; i + 1;
d) i < min; j!= (n+i-k)mod n; A[(j + k)mod n]; temp; i + 1;

16. Consider the following C program that attempts to locate an element x in an array Y[] using binary search. The program is erroneous.

```
 f(int Y[10], int x) {
   int i, j, k;
   i = 0; j = 9;
   do {
        k =  (i + j) /2;
        if( Y[k] < x)  i = k; else j = k;
     } while(Y[k] != x && i < j);
   if(Y[k] == x) printf ("x is in the array ") ;
   else printf (" x is not in the array ") ;
 }
```

On which of the following contents of Y and x does the program fail?

a) Y is [1 2 3 4 5 6 7 8 9 10] and x < 10
b) Y is [1 3 5 7 9 11 13 15 17 19] and x < 1
c) Y is [2 2 2 2 2 2 2 2 2 2] and x > 2
d) Y is [2 4 6 8 10 12 14 16 18 20] and 2 < x < 20 and x is even

17. You are given a list of 5 integers and these integers are in the range from 1 to 6. There are no duplicates in list. One of the integers is missing in the list. Which of the following expression would give the missing number. **^** is bitwise XOR operator. **~** is bitwise NOT operator. Let elements of list can be accessed as list[0], list[1], list[2], list[3], list[4]

a) list[0] ^ list[1] ^ list[2] ^ list[3] ^ list[4]
b) list[0] ^ list[1] ^ list[2] ^ list[3] ^ list[4] ^ 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6
c) list[0] ^ list[1] ^ list[2] ^ list[3] ^ list[4] ^ 1 ^ 2 ^ 3 ^ 4 ^ 5
d) ~(list[0] ^ list[1] ^ list[2] ^ list[3] ^ list[4])

18. The following C function takes a simply-linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.

```
typedef struct node
```

```
{
  int value;
  struct node *next;
}Node;
Node *move_to_front(Node *head)
{
  Node *p, *q;
  if ((head == NULL: || (head->next == NULL))
    return head;
  q = NULL; p = head;
  while (p-> next !=NULL)
  {
    q = p;
    p = p->next;
  }
  return head;
}
```

a) q = NULL; p->next = head; head = p;
b) q->next = NULL; head = p; p->next = head;
c) head = p; p->next = q; q->next = NULL;
d) q->next = NULL; p->next = head; head = p;

19.
```
void fun(int *p)
{
  int q = 10;
  p = &q;
}
  int main()
{
  int r = 20;
  int *p = &r;
  fun(p);
  printf("%d", *p);
  return 0;
}
```

a) 10
b) 20
c) Compiler error
d) Runtime Error

20.
```
#include <stdio.h>
int main()
{
   int a[5] = {1,2,3,4,5};
   int *ptr = (int*)(&a+1);
   printf("%d %d", *(a+1), *(ptr-1));
   return 0;
}
```

a) 2 5
b) Garbage Value
c) Compiler Error
d) Segmentation Fault

21.
```
#include <stdio.h>
char *c[] = {"AptiTude", "MCQ", "TEST", "GAME"};
char **cp[] = {c+3, c+2, c+1, c};
char ***cpp = cp;
int main()
{
```

```
   printf("%s ", **++cpp);
   printf("%s ", *--*++cpp+3);
   printf("%s ", *cpp[-2]+3);
   printf("%s ", cpp[-1][-1]+1);
   return 0;
}
```

a) TEST iTude E CQ
b) MCQ Tude E CQ
c) TEST Tude E CQ
d) *GarbageValue* iTude E CQ

22. Predict the output

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
void fun(char** str_ref)
{
   str_ref++;
}
 int main()
{
   char *str = (void *)malloc(100*sizeof(char));
   strcpy(str, "Computers");
   fun(&str);
   puts(str);
   free(str);
   return 0;
}
```

a) Computers
b) omputers
c) Garbage Value
d) Compiler Error

23. Assume that the size of int is 4.

```
#include <stdio.h>
void f(char**);
int main()
{
   char *argv[] = { "ab", "cd", "ef", "gh", "ij", "kl" };
   f(argv);
   return 0;
}
void f(char **p)
{
   char *t;
   t = (p += sizeof(int))[-1];
   printf("%sn", t);
}
```

a) ab
b) cd
c) ef
d) gh

24. Consider the following C declaration

```
struct {
   short s[5];
   union {
      float y;
      long z;
```

```
    }u;
} t;
```
Assume that objects of the type short, float and long occupy 2 bytes, 4 bytes and 8 bytes, respectively. The memory requirement for variable t, ignoring alignment considerations, is

a) 22 bytes
b) 14 bytes
c) 18 bytes
d) 10 bytes

25.
```
#include<stdio.h>
struct st
{
   int x;
   struct st next;
};
int main()
{
   struct st temp;
   temp.x = 10;
   temp.next = temp;
   printf("%d", temp.next.x);
   return 0;
}
```
a) Compiler Error
b) 10
c) Runtime Error
d) Garbage Value

26. Consider the program below in a hypothetical language which allows global variable and a choice of call by reference or call by value methods of parameter passing.
```
int i ;
program main ()
{
   int j = 60;
   i = 50;
   call f (i, j);
   print i, j;
}
procedure f (x, y)
{
   i = 100;
   x = 10;
   y = y + i ;
}
```
Which one of the following options represents the correct output of the program for the two parameter passing mechanisms?

a) Call by value : i = 70, j = 10; Call by reference : i = 60, j = 70
b) Call by value : i = 50, j = 60; Call by reference : i = 50, j = 70
c) Call by value : i = 10, j = 70; Call by reference : i = 100, j = 60
d) Call by value : i = 100, j = 60; Call by reference : i = 10, j = 70

27. In fopen(), the open mode "wx" is sometimes preferred "w" because. 1) Use of wx is more efficient. 2) If w is used, old contents of file are erased and a new empty file is created. When wx is used, fopen() returns NULL if file already exists.

a) Only 1
b) Only 2
c) Both 1 and 2
d) Neither 1 nor 2

28. The output of following C program is

```
#include <stdio.h>
char str1[100];
char *fun(char str[])
{
   static int i = 0;
   if (*str)
   {
      fun(str+1);
      str1[i] = *str;
      i++;
   }
   return str1;
}
int main()
{
   char str[] = "COSSCO Aptitude Test 2019";
   printf("%s", fun(str));
   return 0;
}
```

a) COSSCO Aptitude Test 2019
b) 9102 tset edutitpA OCSSOC
c) Nothing is printed on screen
d) Segmentation Fault

29. Consider the following function written in the C programming language. The output of the above function on input “ABCD EFGH” is

```
void foo (char *a)
{
 if (*a && *a != ` `)
 {
   foo(a+1);
   putchar(*a);
 }
}
```

a) ABCD EFGH
b) ABCD
c) HGFE DCBA
d) DCBA

30. What is the output of following program?

```
#include <stdio.h>
#define macro(n, a, i, m) m##a##i##n
#define MAIN macro(n, a, i, m)

int MAIN()
{
   printf("COSSCO2019");
   return 0;
}
```

a) Compiler Error
b) COSSCO2019
c) MAIN
d) main