QUIZ-I (Data Structures Lab – CSC204)

20je0735@cse.iitism.ac.in Switch account

Your email will be recorded when you submit this form

QUIZ-I (Data Structures Lab – CSC204)

i. This Quiz Test contains 15 questions with total marks of 15.
ii. Answer all the questions.
iii. Duration of the test is 35 Mins.

1 point

Let assume that a stack is supporting ***k_Pop*** operation in addition with all other conventional operations. Implementation of ***k_Pop*** operation is given below. What will be the content of the stack if the following sequence of operations is applied to an initially empty stack?

Push(2)→Push(9)→Push(5)→k_Pop(5,top)→Push(6)→Push(8)→Pop()→Push(5)→Push(3)→Push(8)→ Pop()→Push(7)→Push(3)→k_Pop(3,top)

```
void k_Pop(k, top){
if ( isEmpty() == True )
        print( "Stack is empty!" )
else if (k <= top + 1)
        top = top - k
else
        top = −1
}
```

- [ ] 6 5 3 7
- [ ] 6 5
- [ ] 2 9 6 8 5 3 7
- [ ] 2 9 5 6 5 3 7
- [ ] 6 5 3

1 point

Two matrices P and Q can be stored in two different arrays $arr1$ and $arr2$, respectively. Either of row-major order or column-major order can be used to store the elements in $arr1$ and $arr2$ using contiguous memory locations. Now, which of the following option is most suitable corresponding to the time complexity of an algorithm to compute multiplication operation $(P \times Q)$?

- [ ] Independent of the storage scheme.
- [ ] Least complexity if both are in column-major order.
- [ ] Least complexity if both are in row-major order.
- [ ] Matrices cannot be stored in row-major or column-major order.
- [ ] Least complexity if arr1 is in row-major, and arr2 is in column-major order.
- [ ] High complexity as number of multiplications and addition get increased in row-major or column-major representation.
- [ ] None of these.
- [ ] Least complexity if arr1 is in column-major order, and arr2 is in row-major order.

1 point

Consider the following C program, expected output of this C program is **"824, 6, 824"**. Among the given options which code segment(s) can be used in place of $\boldsymbol{A}, \boldsymbol{B}$, and $\boldsymbol{C}$ to get the desired output. Assume that each integer required four bytes of memory location, and the base address of array p is 800.

```
int main(){
int p[4][3] = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};
printf(%u,%u,%u, A, B, C);
}
```

$A = p + 2$
$B = p[1][2]$
$C = p + 2$

☐ Option 4

$A = p + 2$
$B = *(*(p + 1) + 2)$
$C = *(p + 1) + 3$

☐ Option 1

$A = *(p + 1) + 3$
$B = p[1][2]$
$C = p + 2$

☐ Option 3

$A = p + 3$
$B = *(p + 2) + 3$
$C = *(p + 3)$

☐ Option 2

All the above

- [ ] Option 5

1 point

$A\,[1 \ldots n]$ is a one-dimensional array in C, which of the following statement(s) is/are true? Select the correct option(s).

S1: $A + 1$ is not an illegal operation.
S2: $A + +$ is not a legal operation.
S3: Maximum three-dimensional array is allowed in C Language.
S4: It is mandatory to initialize an array while declaring.
S5: $A[i]$ is same as $* (A + i)$

- [ ] S1
- [ ] S3
- [ ] S4
- [ ] S1, S5
- [ ] S2, S4
- [ ] S2, S5
- [ ] S3, S5

1 point

Which of the following statement(s) is/are false? Choose the appropriate option(s).

S1: The linked lists require data movements during *insertion* and *deletion* operations.

S2: The first row in *sparse matrix representation* contains number of rows, number of zeros, and number of non-zero terms of a *sparse matrix*.

S3: Arrays are suitable for the applications where sequential access is required and *insertion/deletion* operations are performed at the end.

S4: During *deletion* operation in the linked list, the next pointer of the item immediately preceding the one to be deleted is altered, and made to point to the item following the deleted item.

S5: The set of *axioms* describe the syntax of the operations in a data structure.

- [ ] S2, S3, S5
- [ ] S2, S3, S4
- [ ] S3, S4
- [ ] S1, S3, S4
- [ ] S1, S2, S5
- [ ] S1, S3, S5
- [ ] S1, S2, S3
- [ ] S1, S4

1 point

Consider the following C code and select the correct option(s) which will print the value of 5.

```
int x[10][20][30] = {0};
x[5][2][1] = 5;
```

- [ ] None of the option
- [ ] Option `printf("%d",*(((x+5)+2)+1));`
- [ ] Option `printf("%d",***((x+5)+2)+1);`
- [ ] Option `printf("%d",*(*(*(x+5)+2)+1));`

1 point

You have been given following function to perform reversal operation in the doubly linked list:

```
struct node * reverse (struct node * start)
 {
       struct node  * p1,* p2;
       p1 = start;
       p2 = p1 → next;
       p1 → next = NULL;
       p1 → prev = p2;
       while (p2! = NULL)
       {
             ----------------------
             ----------------------
             ----------------------
             p2 =  p2 → prev;
       }
       start = p1;
       printf("List reversed \n");
       return start;
 }
```

Some of the lines in the above code are missing. Choose the correct lines of codes from the following options which can be utilized at the missing locations to rightly complete the above given function for the requisite purpose.

```
p2 → next = p1 → prev;
p1 → next = p1;
p2 = p1;
```

☐ Option 5

```
p2 → prev = p1 → next;
p1 → next = p1;
p2 = p1;
```

☐ Option 2

```
p2 → prev = p2 → next;
```

```
p1 → prev = p2 → next;
```

$p2 \rightarrow next = p1;$
$p1 = p2;$

Option 4

$p2 \rightarrow next = p1;$
$p1 = p2;$

Option 1

$p2 \rightarrow prev = p1 \rightarrow prev;$
$p1 \rightarrow next = p1;$
$p1 = p2;$

Option 6

$p1 \rightarrow next = p2 \rightarrow next;$
$p1 \rightarrow next = p1;$
$p2 = p1;$

Option 3

1 point

Which of the following statement(s) is/are false? Choose the appropriate option(s).

S1: During *insertion* operation in a singly linked list, the *link* pointer of the new record is set to link it to the item which is to precede it in the list.

S2: A best algorithm can be judged by its time complexity only.

S3: Deleting an element somewhere in the middle of the array would require each subsequent element to be moved one location upward.

S4: An algorithm satisfies its criteria if each instruction is clear, unambiguous, and basic.

S5: During *insertion* operation in the middle of the singly linked list, two link pointers need to be modified.

S6: A two-dimensional array can be assumed as a single column with many rows and mapped sequentially. Such representation is known as *column-major order*.

- [ ] S1
- [ ] S3, S4, S5
- [ ] S1, S3, S4
- [ ] S1, S4
- [ ] S2, S4
- [ ] S1, S2
- [ ] S1, S2, S6
- [ ] S2, S3, S5

Consider the following function and select the appropriate option(s): 1 point

```
void fn( struct node * head)
{
 if (head == NULL)
  return;
   fn(head-> next);
   printf("%d",head-> data);
}
```

- [ ] It only displays nodes at odd positions
- [ ] It only displays nodes at even positions
- [ ] It only displays the first node of linked list
- [ ] It only displays last node of linked list
- [ ] It displays all nodes of the linked list starting from first node
- [ ] It displays nodes in the reverse order

1 point

Consider the following function ***compute*** defined below:

```
struct object
{
int value;
struct object * link;
};

int compute(struct object * a)
 {
        return ((a == NULL)||(a → link == NULL)||
                ((a → value ≤ a → link → value)&&compute(a → link)));
}
```

For the above linked list, the function ***compute*** returns 1 if and only if

- [ ] Linked list contains exactly three elements.
- [ ] Elements of the linked list are sorted in increasing order of the value.
- [ ] Linked list is empty.
- [ ] All the elements of the linked list are distinct.
- [ ] Linked list is non-empty and it contains exactly two elements.
- [ ] Elements of the linked list are not sorted in decreasing order of the value.
- [ ] Elements of the linked list are sorted in non-decreasing order of the value.
- [ ] Linked list contains exactly one element.

1 point

Suppose A is an array that contains integers in increasing order. Let the following code determines that there are elements in the array whose difference is D $(D > 0)$. Find the correct code for exp.

```
i = 0;
j = 0;
while(j < N)
{
if(exp) j++;
elseif(A[j] − A[i] == 0) break;
else i++;
}
if(j < N)
printf("yes");
else
printf("No");
```

- [ ] $A[i] + A[j] < D$ — Option 5
- [ ] $A[j] - A[i] < D$ — Option 2
- [ ] $A[j] - A[i] > D$ — Option 1
- [ ] $A[i] - A[j] < D$ — Option 4

$A[i] + A[j] > D$

Option 6

$A[i] - A[j] > D$

Option 3

1 point

You have been provided a two-dimensional array P of integers which is declared as: $P : array[1 \ldots 20][1 \ldots 30];$

If the given array P is stored using *row-major order* assuming that each integer takes one memory location (where first element of the array is considered to be stored at location 100), what will be the address of the element $P[x][y]$?

- 30y+15x+60
- None of these
- 30y+x+69
- 20x+12y+84
- 30x+y+69
- 15x+12y+84
- 40x+12y+84
- 45y+12x+80

1 point

Which of the following task(s) can be performed successfully? Assume that only the head pointer of the linked list is available to perform all these tasks. Select all the correct options.

T1: Determining the middle element of a linked list in a single pass.

T2: Converting a singly linked list into a circular linked list in constant time.

T3: Deleting the n^{th} node from the end of a singly linked list in a single pass.

T4: Deleting the last element of a circular doubly linked list without traversing the whole list.

- [ ] T2
- [ ] T2, T3, T4
- [ ] T1, T2, T4
- [ ] T2, T3
- [ ] T4
- [ ] T1
- [ ] T3, T4

1 point

Suppose, we have been provided two singly linked lists and we want to append one at the end of the another where $start1$ and $start2$ points to the first node of the linked lists. To perform above task, following code is given:

```
ptr → link = start2;
struct node * concat (struct node * start1, struct node * start2)
{
        struct node  * ptr;
        if (start1 == NULL)
        {
                start1 = start2;
                return start1;
        }
        if (start2 == NULL)
                return start1;
        ptr = start1;

        ----------------------
        ----------------------
        ----------------------
        return start1;

}
```

Which of the following option(s) correctly represent the missing lines of codes for completion of the above given function to fulfill its purpose?

```
while(ptr → link! = NULL)
ptr = ptr → link;
ptr → link = start1;
```

Option 1

```
while(ptr → link! = NULL)
ptr → link = ptr;
ptr = start1;
```

Option 5

```
while(ptr → link! = NULL)
```

```
while(ptr → link! = NULL)
```

$ptr = ptr \rightarrow link;$
$ptr \rightarrow link = start2;$

Option 3

$ptr \rightarrow link = ptr$
$ptr = start2;$

Option 2

$while(ptr \rightarrow link == NULL)$
$ptr = ptr \rightarrow link;$
$ptr = start1;$

Option 4

$while(ptr \rightarrow link == NULL)$
$ptr \rightarrow link = ptr;$
$ptr = start1;$

Option 6

1 point

$\boldsymbol{A}[\,] =$ "$\boldsymbol{Char1}$" and $\boldsymbol{B}[\,] =$ "$\boldsymbol{Char2}$" are two-character array. What does the invoking of $\boldsymbol{fun}(\boldsymbol{A}, \boldsymbol{B})$ will do?

```
void fun(char * s, char * t)
{
   while(* s ++ = * t ++);
   }
```

A. Overwrite the elements of array $\boldsymbol{A}$ up to index 3, by the corresponding elements of array $\boldsymbol{B}$.
B. All the elements of array $\boldsymbol{A}$ will be overwritten by the corresponding elements of array $\boldsymbol{B}$.
C. There will be segmentation fault, because the program will try to access the array elements beyond the limit.
D. There will be compilation error, because this is an invalid assignment of array elements.

D

C

A

B

Send me a copy of my responses.

Back Submit Clear form