Week 5 Solutions

1.Which of the following is not correct with respect to an Abstract Data Type (ADT)?

A) There can be several implementations of the same ADT.

B) ADT encapsulates the representation of data

C) ADT has two components : data structure and operations defined on the data structure.

D) Use of a ADT is dependent upon the implementation

Answer D)

ADT does not depend on the implementation

2. Which of the following is correct about the operator overloading in C++?

A) Overloading of all built-in operators are supported.

B) Arity of the operators cannot be changed.

C) Precedence and associativity of the operators can be changed.

D) None of the above

Answer B)

Overloading of built-in operators like ternary (?:) and scope resolution (::) are not possible

Arity, precedence and associativity of operators cannot be changed.

3.A set is represented using the List ADT done in class. Which of the operations will be the slowest?

A) union

B) intersection

C) membership

D) cardinality

Answer A)

Finding membership and cardinality will require a linear scan.

Between union and intersection, union will require more operations than intersection as union will have to scan both Lists and include only the unique elements in the union.

4.Interpret what the following function computes given head nodes of two list.

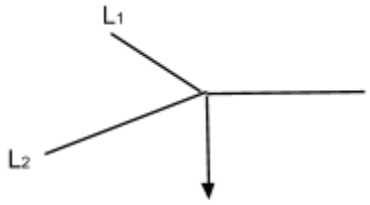
```
<pre>
struct LinkedList{
   int value;
   LinkedList* next;
}
LinkedList* DoSomething(LinkedList* first, LinkedList* second){
  int len1 = length(first);
  int len2 = length(second);
  int diff;  LinkedList* tmp1,*tmp2;
  if (len1 > len2){
    diff = len1-len2;
    tmp1 = first; tmp2 = second;
  }else{
    diff = len2-len1;
    tmp1 = second; tmp2 = first;
  }
  while(diff>0){
    tmp1=tmp1->next;
    diff--;
  }
```

```
 while(tmp1!=NULL &&
         tmp2!=NULL &&
         tmp1!=tmp2){
   tmp1=tmp1->next;
   tmp2=tmp2->next;
 }
 if (tmp1==NULL)
   return NULL
 else
   return tmp1;
}
</pre>
```

length() computes the length of linked-list.

A) Find the node where two list have same value and return NULL if there is no such element.
B) Find the node where two list intersects and return NULL if there is no intersection.
C) Find the node in the longest list, at length of smallest list and return NULL if length of two list are same.
D) None of the

Answer B)
Explanation:
diff computes the difference between length of two list.
Then move the head of longer by diff.
Follow it by moving the pointers of both list till they match.



Idea is to find this intersection point of the 2 linked lists (L1 & L2)

5.
Which of the following is true about stack ADT implemented using linked list ?

A. For push( ) operation, new nodes are inserted at the head of linked list. For pop( ) operation, nodes are removed from head.
B. push( ) adds new node at head of list and pop( ) removes nodes from rear end of list
C. Push() adds new node at rear end of list and Pop() removes nodes from head of list.
D. None of above

Answer A)

6.
Convert the following infix expression to postfix expression. (^ is the exponentiation operator)

(a/b^c)*b*c-d*e

A. ab/c^b*c*de*-
B. abc^/bc**de*-
C. abc/^b*c*de-
D. abc^/bc*de**-

Answer B)
Explanation
(a/b^c)*b*c-d*e =

(a/(b^c))*(b*c)-(d*e)

7. Suppose a circular queue is implemented with an array of n elements. What is the condition for a queue being full?

A. REAR == FRONT
B. (REAR+1) mod n == FRONT
C. (FRONT+1) mod n == REAR
D. None of Above

Answer B)

8.Given two structures tmp1, tmp2 as defined below, how will you compare the two structures?

struct A { int a; };
struct A tmp1,tmp2

A. tmp1==tmp2
B. tmp1.a==tmp2.a
C. Both A & B
D. tmp1 and tmp2 cannot be compared.

Answer B)

9. A queue is a _____
A. FILO
B. LIFO
C. FIFO
D. None of the above.

Answer C)

10. In general, List ADT allows:
A. Insertions and deletions anywhere.
B. Insertions and deletions only at one end.
C. Insertions at back and deletions at the front.
D. Insertions at the front and deletions at the back,

Answer A)

11. A dynamic array is to be created in C++. Which of the statements will achieve this?

A. int *A = new int [5];
B. int *A= new int*[5];
C. int *A = new int(5);
D. int A = new int[5];

Answer A)

12.What does the following function DoSomething( ) do on the linked list head (passed as argument to function? Assume that Node is a structure that has an integer member called *value* and a pointer to another.

```
struct Node {
   int value;
   Struct Node * next;
};

void DoSomething (struct Node *head)
 {
     if (head == NULL) return;
    else {
          DoSomething  (head->next);
          cout << head->value;
    }
}
```

A. print values in linked list in order
B. print alternate values in linked list
C. print values in linked list in reverse order
D. Error.

Answer C)

13.Suppose C supported a hypothetical data-type called **digit** which could only hold a single **digit** (0 - 9). You are needed to write a program that adds two digits. You soon realize that sum of 2 digits can exceed 9 and hence one single **digit** variable cannot hold the number. To rectify this problem you define a structure digit2 (2 digits) as follows.

```
struct digit2 {
   digit unit;  /* digit variable that holds the units place digit */
```

```
    digit tens;  /* digit variable that holds the tens place digit */
};
typedef struct digit2 Digit2; /* avoid frequent use of struct keyword */
```

The task now is to write the code that multiplies the two digits.

```
Digit2 Product (digit a, digit b) {
        Digit2 prod;
        if ( a == 1)
         prod.tens = 0;
         prod.unit = b;
        … /* similar code for b = 1 */
        /* Rest of the logic that covers other cases */
 return prod;
}
```

You then need to now write a function that adds up two digits. You decide to reuse the product function rather than write an add function. (May not necessarily be a good idea).What is the flaw (if any) with this implementation ?

```
Digit2 Sum(digit a, digit b) {
    digit i;
    Digit2 sum;
        for (digit i = 0; i < b; i++) {
           sum = sum + Product(1, b);
        }
        return sum;
}
```

A. There is no flaw.
B. Incorrect because of a logical error.
C. Incorrect because one cannot add two struct variables directly.

Answer: **C**

One cannot add two struct variables directly. Need to do component wise addition.

14. A node in a linked list is defined as:

```
struct node {
   int data;
   struct node* next;
}
```

List has elements 4->5->3->2->1->2.

The following function is executed.

```
void print(struct node* head, int flag){
   while(head != NULL){
         if(flag){
                head = head->next;
                printf("%d\t",head->data);
         }
         else{
                printf("%d\t",head->data);
                head = head->data;
         }
   }
   return;
}
```

Variable *head* stores the pointer to the head of the list.

What is the output of print(head,1)?

A) 4 3 1

B) 5 2 2

C) 4 5 3 2 1 2

D) 5 3 2 1 2

Answer: D

Explanation: When the flag variable is set print() function skips the head.

15. A node in a linked list is defined below

```
struct node
{
    int data;
    struct node* next;
}
```

List has elements 4->5->3->2->1->2.

```
int count(struct node *head){
    int count = 0;
    if (head == NULL)
            return count;
    while(head->next != NULL){
            count++;
            head = head->next;
    }
    return count;
}
```

variable head stores the pointer to the head of the list.

What is the value returned by count(head) ?

1) 6

2) 5

3) 0

4) 4

Answer: B

Explanation: The way the code is written skips counting one element.

16.

```
 struct point{
        int x,y;
};
typedef struct point Point;
struct ptlist{
        Point pt;
        struct ptlist* next;
};
struct plist *head;
```

head is a pointer to the head of a linked list of points that is built. How do you access the member x of the head node of the list?

A. head->pt->x

B. head->pt.x

C. Both are equivalent

D. Both are incorrect

Answer. **B)**

A pointer to a structure lets access to the member variables using “->” operator and a normal variable lets access through the “.” operator.