|  |  |
| --- | --- |
| 1. Which of the following data structure can't store the non-homogeneous data elements?  a. Arrays b. Records c. Pointers d. None | a. Arrays |
| 2. Which of the following data structure store the homogeneous data elements?  a. Arrays b. Records c. Pointers d. None | b. Records |
| 3. Each data item in a record may be a group item composed of sub-items; those items which are indecomposable are called  a. elementary items b. atoms  c. scalars d. all of above | d. all of above |
| 4. The difference between linear array and a record is  a. An array is suitable for homogeneous data but hte data items in a record may have different data type  b. In a record, there may not be a natural ordering in opposed to linear array.  c. A record form a hierarchical structure but a lienear array does not  d. All of above | d. All of above |
| 5. Which of the following statement is false?  a. Arrays are dense lists and static data structure  b. data elements in linked list need not be stored in adjecent space in memory  c. pointers store the next data element of a list  d. linked lists are collection of the nodes that contain information part and next pointer | c. pointers store the next data element of a list |
| 6. A data structure where elements can be added or removed at either end but not in the middle  a. Linked lists b. Stacks c. Queues d. Deque | d. Deque |
| 7. When new data are to be inserted into a data structure, but there is no available space; this situation is usually called  a. underflow b. overflow c. housefull d. saturated | b. overflow |
| 8. The situation when in a linked list START=NULL is  a. underflow b. overflow c. housefull d. saturated | a. underflow |
| 9. Which of the following is two way list?  a. grounded header list b. circular header list  c. linked list with header and trailer nodes d. none of above | d. none of above |
| 10. Which of the following name does not relate to stacks?  a. FIFO lists b. LIFO list  c. Piles d. Push-down lists | a. FIFO lists |
| 11. The term "push" and "pop" is related to the  a. array b. lists c. stacks d. all of above | c. stacks |
| 12. The operation of processing each element in the list is known as  a. Sorting b. Merging c. Inserting d. Traversal | d. Traversal |
| 13. Finding the location of the element with a given value is:  a. Traversal b. Search c. Sort d. None of above | b. Search |
| 14. Linked lists are best suited  a. for relatively permanent collections of data  b. for the size of the structure and the data in the structure are constantly changing  c. for both of above situation  d. for none of above situation | b. for the size of the structure and the data in the structure are constantly changing |
| 15. Which of the following is *not* a dynamic data structure?  a. Linked list. b. Stack.  c. Array. d. Binary tree. | c. Array. |
| 16. In general, linked lists allow:  a. Insertions and removals anywhere.  b. Insertions and removals only at one end.  c. Insertions at the back and removals from the front.  d. None of the above. | a. Insertions and removals anywhere. |
| 17. Which data structure represents a waiting line and limits insertions to be made at the back of the data structure and limits removals to be made from the front?  a. Stack. b. Queue. c. Binary tree. d. Linked list. | b. Queue. |
| 18. What kind of linked list begins with a pointer to the first node, and each node contains a pointer to the next node, and the pointer in the last node points back to the first node?  a. Circular, singly-linked list. b. Circular, doubly-linked list.  c. Singly-linked list. d. Doubly-linked list. | a. Circular, singly-linked list. |
| 19. How many pointers are contained as data members in the nodes of a circular, doubly linked list of integers with five nodes?  a. 5 b. 8 c. 10 d. 15 | c. 10 |
| 20. Which of the following statements about stacks is incorrect?  a. Stacks can be implemented using linked lists.  b. Stacks are first-in, first-out (FIFO) data structures.  c. New nodes can only be added to the top of the stack.  d. The last node (at the bottom) of a stack has a null (0) link. | b. Stacks are first-in, first-out (FIFO) data structures. |
|  |  |
|  |  |

**Linked List Exercises**

*Note* All linked lists are null-terminated and singly linked. Input lists may be empty, unless otherwise stated. If new nodes may be created, then report whether or not space was exhausted, using a boolean reference parameter.

Answer each question with a C++ method (function).

1. **Insert in order** Given a linked list of integers sorted from smallest (at the head end) to largest, and a pointer to a single node containing an integer, insert the node in the linked list so that it remains sorted.
2. **Cumulative sum** Given a null-terminated linked list, in, create a new null-terminated linked, list out, of the same length, such that node*i* of out contains the sum of the data in in's nodes up to and including node*i* of list in. Detect heap exhaustion and report it by setting a boolean variable.
3. **Delete last node** Given a nonempty list, delete the last node and set the new last link to null.
4. **Deal** Given a null terminated linked list, rearrange its nodes into two lists: <first node, third node, fifth node, ...> and <second node, fourth node, sixth node, ...>. Do not allocate any new nodes.
5. **Rifle Shuffle**Given two null terminated linked lists, combine their nodes so that the nodes of the new list alternate between those of the two original nodes: <first node of first list, first node of second list, second node of first list, second node of second list, ... >. Do not allocate any new nodes.
6. **Catenate** Given two null-terminated linked lists headed by **left** and **right**, set the last link of the left list to point to the right list, thus joining them into one list. Do not allocate any new nodes.
7. **Reverse** Given a null-terminated linked lists headed reverse the order of its nodes. Do not allocate any new nodes.
8. **Read a list** Given a nonnegative integer *i*, read *i* integers from the input and build a null-terminated linked list such that the first integer in the input is the first node and so forth. Detect heap exhaustion and report it by setting a boolean variable.
9. **List to int** Given a null-terminated linked list of integers from 0 to 9 (inclusive) representing a nonnegative integer in decimal (least significant digit at the head), compute into an **unsigned** variable, the integer that it represents. (The empty list represents 0.) You may assume that the list represents a number that can be represented by an **unsigned**.
10. **Int to list** Given an non-negative integer , create a null-terminated linked list of integers between 0 and 9 representing the integer (least significant digit first). (0 is represented by an empty list.) Set a boolean variable to indicate whether or not space has been exhausted.

**Hint**. In C++ **i/10**, where i is an integer expression, gives the integer quotient of i divided by 10, and **i%10** gives the remainder.

**11)**  Consider the List class with the following private members:

class List{

/\* public members here ... \*/

private:

struct Node{

ListDataType item; // the data of the node

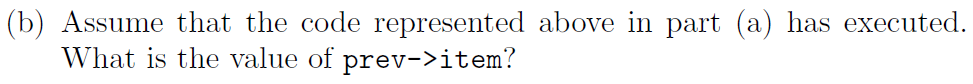
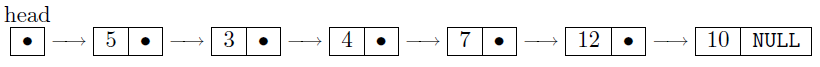
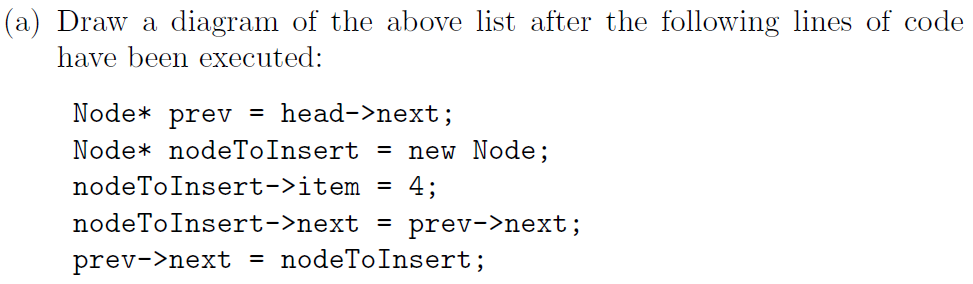
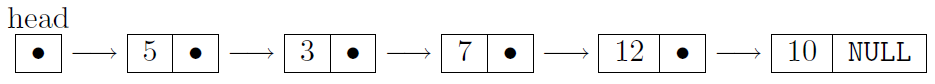
Node\* next; // points to the next node of the list

};

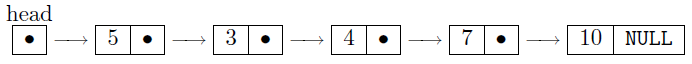
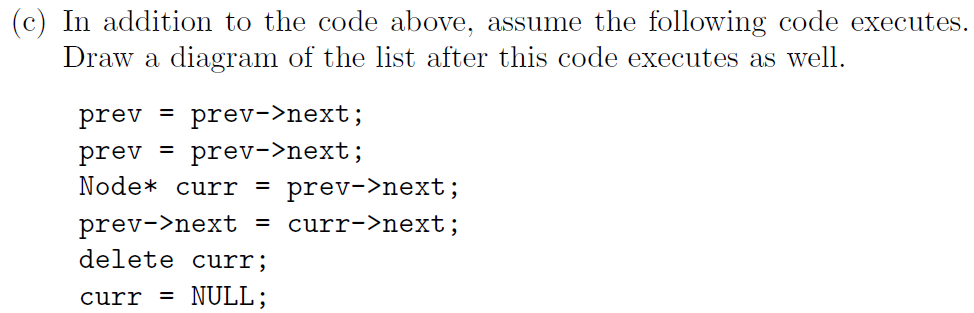
Node\* head; // point to first node in the list

};

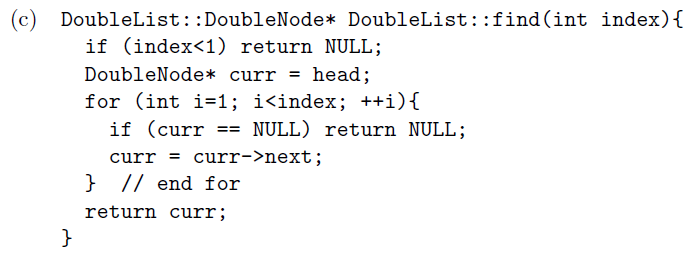
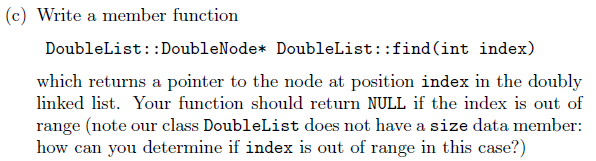
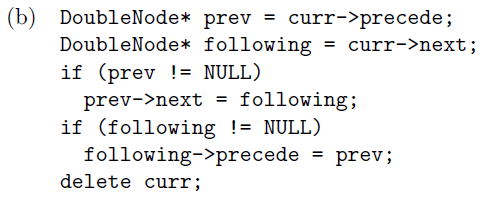
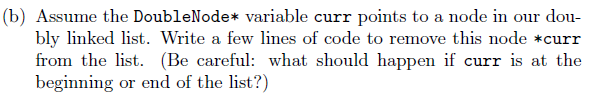
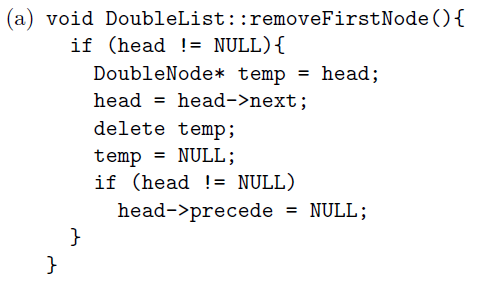
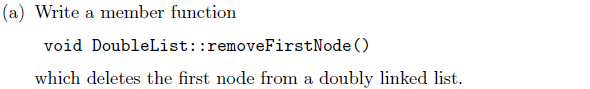
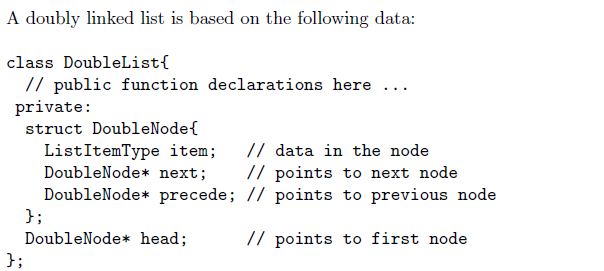
Consider the linked list of ints represented by the following diagram:



3



**12)**



**Stack Exercises**

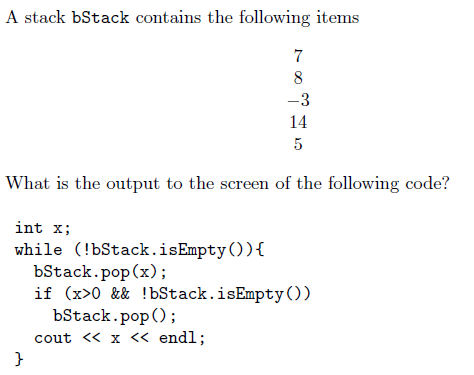
**1)**

Output

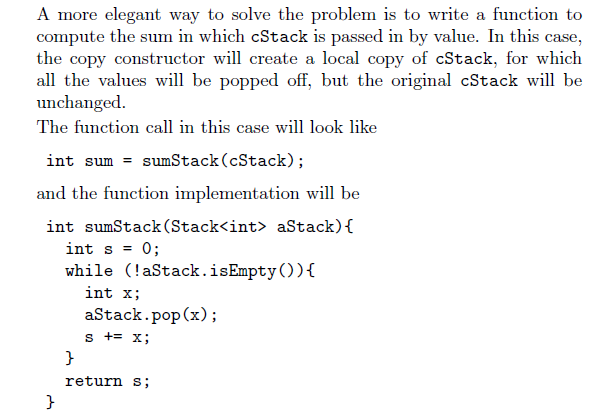
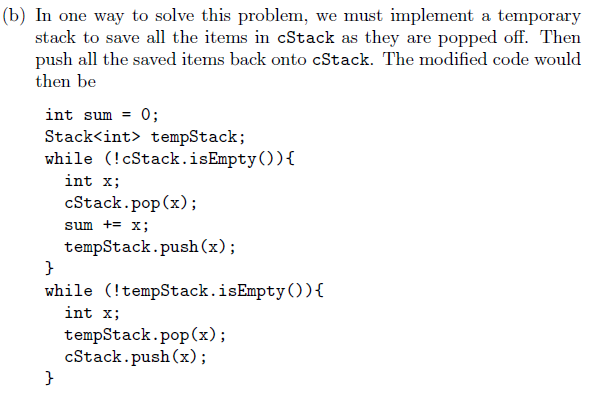
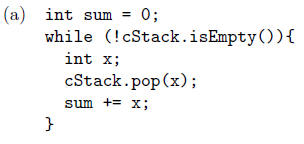
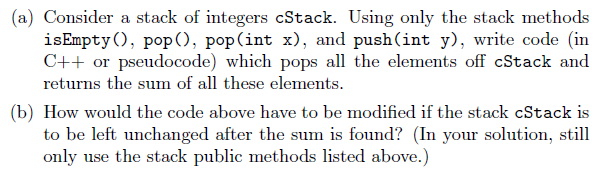
7

-3

14



**2)**



**3)**

(a) If a = 10, b = 3, c = -2, evaluate the postfix expression aa+bc-\*

(b) Convert the following fully parenthesized infix expression to postfix.

(x-((x+y)\*((z/r)+p)))

(a) aa+bc-\* is (20)bc-\* is (20)5\* is 100,

since aa+ is 10+10 = 20,

bc- is 3 - (-2) = 5 and (20)5\* is 20 \* 5 = 100.

(b) xxy+zr/p+\*-

**Queue Exercises**

**1)**

