Jonathan Sum

|  |  |
| --- | --- |
| 1. What is the output produced by the following code? *int* \*p1, \*p2; p1 = *new int*; p2 = *new int*; \*p1 = 10; \*p2 = 20; cout << \*p1 << " " << \*p2 << endl; p1 = p2; cout << \*p1 << " " << \*p2 << endl; \*p1 = 30; cout << \*p1 << " " << \*p2 << endl;   10 20  20 20  30 30  How would the output change if you were to replace the statement \*p1 = 30; with the following? \*p2 = 30;  30 30 | 1. What is the output of the following C++ code? int x; int y; int \*p = &x; int \*q = &y; \*p = 35; \*q = 98; \*p = \*q; cout << x << " " << y << endl; cout << \*p << " " << \*q << endl;   98 98  98 98 |
| 1. What is the output of the following C++ code? int \*p; int \*q; p = new int; q = new int; \*p = 27; \*q = 35; cout << \*p << " " << \*q << endl; \*q = \*p; \*p = 73; cout << \*p << " " << \*q << endl; p = new int; \*p = 36; q = p; cout << \*p << " " << \*q << endl;   27 35  73 27  36 36 | 1. What is the output of the following code? int \*secret; int j; secret = new int[ 10 ] ; secret [ 0 ] = 10; for (j = 1; j < 10; j++)   secret [j] = secret [ j - 1 ] + 5; for (j = 0; j < 10; j++)  cout << secret [ j ] << " "; cout << endl; 10 15 20 25 30 35 40 45 50 55 |
| 1. Consider the following function: int test( int x, int y) { if (x == y)   return x; else if (x > y)  return (x + y); else  return test(x + 1, y - 1); } What is the output of the following statements? a. cout << test(5, 10) << endl;  15  b. cout << test(3, 9) << endl;  6 | 1. Consider the following C++ statements.  IntList list; list.push\_front(15); list.push\_back(28); list.push\_front(30); list.push\_front (2); list.push\_back(45); list.push\_front (38); list.push\_back (25); list.pop\_front(); list.push\_front (18); list.pop\_front (); list.pop\_front (); list.display(); What is the output of this program segment?   30 15 28 45 25 |
|  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1. Consider the following recursive function: void recFun(int x) { if (x > 10) {   recFun(x / 10);  cout << x % 10 << endl; } else  cout << x << endl; }  2  5  8 | | | What is the output of the following statements?  a. recFun(258);  b. recFun(7);  7  c. recFun(36);  3  6  d. recFun(-85);  **-85** | | |
|  |  |  |  |

Consider the linked list shown below. Assume that the nodes are in the usual info-link form. Use this list to answer Problems 8 through 9. If necessary, declare additional variables. (Assume that list, p, s, A, and B are pointers of type nodeType. You do not need to declare any more pointer variables)

|  |  |
| --- | --- |
|  | struct NodeType  {  int info;  NodeType\* link;  }; |

|  |  |
| --- | --- |
| 1. What is the value of each of the following relational expressions? a. list->info >= 18   1  b. list->link == A  0  c. A->link->info == 16  0  d. B->link == NULL  0  e. list->info == 18  1 | 1. Write C++ statements to do the following. (Assume the questions are independent of each other.)   a. Make A point to the node containing info 23.  A=A->link;  b. Make list point to the node containing 16. list=list->->link->link->link;  c. Make B point to the last node in the list. B = B->->link->link;  d. Make list point to an empty list. list=NULL  e. Set the value of the node containing 25 to 35. list->info=25;  list->link->info = 35;  f. Create and insert the node with info 10 after the node pointed to by A.  NodeType \*node = new NodeType();  node->info = 10;  list->link=node;  node->link=A;  g. Delete the node with info 23. Also, deallocate the memory occupied by this node.  NodeType temp = list->link->link;  NodeType prev = list->link;  prev->list= temp->link  delete temp; |

|  |  |
| --- | --- |
| 1. Show what is produced by the following C++ code. Assume the node is in the usual info-link form with the info of type int. (list and ptr are pointers of type nodeType.) list = new nodeType; list->info = 10; ptr = new nodeType; ptr->info = 13; ptr->link = NULL; list->link = ptr; ptr = new nodeType; ptr->info = 18; ptr->link = list->link; list->link = ptr;   cout << list->info << " " << ptr->info << " "; ptr = ptr->link; cout << ptr->info << endl; | 10 18 13 |

1. You should be able to code any member function for a doubly linked list.
2. Understand the Big 3/Rule of 3 and be able to define any of those functions.
3. You should know how to overload operators.
   1. As a member function or as a friend function
      1. Understand that some operators can only be overloaded as member functions, and some only as friend functions and some either way.
   2. Overload unary operators such as the prefix and postfix operators

|  |  |
| --- | --- |
| 1. Assume we have the following class definition from the IntVector class. | 1. Assume we have the following struct and class definitions from the IntList class. |
| class IntVector{  private:  unsigned sz;  unsigned cap;  int \*data;  public:  IntVector();  IntVector(unsigned size);  IntVector(unsigned size, int value );  ~IntVector();  void reverseResize();  }; | struct IntNode{  int data;  IntNode \*next;  IntNode( int data ) : data(data), next(0){}  };  class IntList{  private:  IntNode \*head;  IntNode \*tail;  public:  IntList();  ~IntList();  void display() const;  void push\_front( int value );  void push\_back( int value );  void pop\_front();  void select\_sort();  void insert\_sorted(int value);  void remove\_duplicates();  }; |
| Write a definition for a function called reverseResize(), that dynamically resizes (by doubling) the array, however the resized array will be in reverse order. | Write the definition of a member that calculates the mean of the values and creates a new node and adds it as the second to last node in the list. |