**Note: Please type all your answers.**

[1] What is the infix expression for the postfix expression: a b b \* + c d + / e +

(a) (a + b2)/(c+d) + e (b) (a + b)\*b/(c+d) + e

(c) a/b2 \*(c+d) + e (d) (a + b2)/c + d + e (e) none of the above

[2] The object **alist** is a list<int> object. Identify the sequence of operations that would store the following elements in **alist**.

8 6 2 5 7

(a) alist.push\_front(2); (b) alist.push\_back(2); (e) none of the above

alist.push\_front(8); alist.push\_back(7);

alist.push\_front(6); alist.push\_front(8);

alist.push\_back(5); alist.push\_front(6);

alist.push\_back(7); alist.push\_back(5);

(c) alist.push\_front(2); (d) alist.push\_front(6);

alist.push\_back(5); alist.push\_back(7);

alist.push\_front(6); alist.push\_back(5);

alist.push\_front(8); alist.push\_back(2);

alist.push\_back(7); alist.push\_front(8);

[3] Assume the element list **alist** contains the characters in the string "mathematics" and assume **iter** is an iterator. What is the sequence of elements after executing the instructions, if the size of the list is 11?

iter = alist.begin();

iter++;

alist.erase(iter++);

iter++;

alist.erase(iter);

alist.pop\_front();

(a) m a t h e m a t (b) a t h e m a t i

(c) m t e a t i c s (d) t e m a t i c s (e) none of the above

[4] Indicate the running time of each algorithm or code segment:

(a) Finding the minimum of an ***n*** element array (unordered).

(i) O(n2) (ii) O(n) (iii) O(n log2 n) (iv) O(n3)

(b) Outputting the first and last letter in a string

(i) O(n2) (ii) O(n) (iii) O(1) (iv) O(log2 n)

(c) Determining the number of tokens (blocks of non-whitespace characters) in a string

(i) O(n2) (ii) O(n) (iii) O(1) (iv) O(log2 n)

(d) for(i = 1; i < n; i++)

for(j = 1; j < n; j++)

sum += i + j;

(i) O(n2) (ii) O(n) (iii) O(1) (iv) O(n3)

(e) for(i = 1; i < n; i++)

for(j = 1; j < i \* i; j++)

count++;

(i) O(n2) (ii) O(n) (iii) O(1) (iv) O(n3)



Note that 12 + 22 + 33 + … + n2 =

[5] Use the following template function f() for parts (a) and (b)

template <typename T>

void f(T arrA[], int n, T arrB[], int& m, T key)

{

int i, j = 0;

for (i = 0; i < n; i++)

if (!(arrA[i] == key))

arrB[j++] = arrA[i];

m = j;

}

(a) Specify the action of the function f()

(i) Inserts all of the elements in arrA that match the key into arrB.

(ii) Inserts all nonduplicate elements from arrA into intB.

(iii) Inserts all of the elements in arrA that do not match the key into arrB.

(iv) Inserts all duplicate elements from arrA into intB.

(b) Would template function f() execute if template type T were rectangle?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[6]

(a) What is the runtime efficiency for inserting an element at an intermediate position in a vector?

(i) O(1) (ii) O(n) (iii) O(lognn) (iv) O(n2)

(b) What is the runtime efficiency for inserting an element at an intermediate position in a list?

(i) O(1) (ii) O(n) (iii) O(lognn) (iv) O(n2)

[7] What is an equivalent non-recursive version of the following function?

int f(int n){

if (n > 0)

return 1 + f(n-1);

else

return 0;

}

[8] What is the action of the following template function f()?

template <typename T>

T f(const list<T>& alist)

{

list<T>::const\_iterator iter = alist.begin();

T mv = \*iter++;

while (iter != alist.end())

{

if (\*iter > mv)

mv = \*iter;

iter++;

}

return mv;

}

(a) returns the range of data values in the list

(b) returns the minimum of the data values in the list

(c) returns the median of the data values in the list

(d) returns the maximum of the data values in the list

(e) none of the above

[9] Assume that \*p is declared as a pointer to a vector of integers:

vector<int> \*p;

(a) Which statement dynamically allocates a vector object that initially contains 8 elements?

(i) p = new vector<int>[8]; (ii) \*p = vector<int>(8);

(iii) p = new vector<int>(8); (iv) \*p = new vector<int>(8);

(b) Which statement inserts the value 5 at the rear of the vector?

(i) (p->push\_back)(5) (ii) \*p->push\_back(5);

(iii) (\*p.push\_back)(5) (iv) p->push\_back(5);

[10] In the following declaration of the class member function f()

void f(const testClass& obj) const;

Distinguish between the use of the attribute const in the argument list and the attribute const immediately after the argument list.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[11] Determine any syntax errors in the class declaration. Illustrate that you understand an error by giving a reasonable correction next to the statement.

(a) class testClassA

{

private

int iValue = 4;

public

void testClassA (int n);

display();

};

(b) class testClassB

{

private int iValue;

// operation uses m to update the current iValue

public void setData(int m) const;

};

[12] The following is a declaration of the class demoClass. In parts (a) thru (e), a well-intentioned but confused programmer attempts to implement the class using external functions. Help the programmer by identifying each error and editing the code with an obvious correction.

class demoClass{

public:

demoClass (int v = 20); // argument v initializes value

int getValue() const; // return current value

void setValue(int v); // assign v to update current value

void addValue(int v); // add v to the current value

private:

int value;

};

(a) Constructor:

demoClass::demoClass(int v): v(value){}

(b) Constructor (another attempt):

demoClass::demoClass(int v = 20){ value = v; }

(c) setValue():

demoClass::void setValue(int v){ value = v; }

(d) addValue():

void demoClass::addValue(int v): value += v; {}

(e) getValue():

int demoClass::getValue(){ return demoClass.value; }

[13] The following is the API for the inventory class. The data members include the integers stockLevel and numberOnHand.

|  |
| --- |
| CLASS inventory Operations |
| inventory (int sLevel);  The constructor uses sLevel to initialize the stockLevel for the item. The constructor also sets the data member numberOnHand to the initial stock level. |
| bool purchase(int n);  The operation simulates a customer purchasing n items. If the inventory has a sufficient number of items on hand, the operation decreases the number by n and return true; otherwise it returns false.  Precondition: If the inventory does not have a sufficient number of items on hand, the operation displays the message "Insufficient inventory" and returns false. The seller immediately reorders the item so that the number on hand is the stockLevel plus n. This reorder insures that the seller could likely meet the customer's demand if the same purchase request is made a short time later.  Postcondition: The number on hand is reduced by n (if the precondition is met) or the number on hand is stockLevel + n if a reorder occurs.  int getNumberOnHand() const;  Returns the current value for number of items on hand. |

(a) Give the declaration for the inventory class.

(b) Implement the constructor using inline code with an initialization list.

(c) Implement the function purchase() using inline code.

(d) Trace the following code and indicate the output.

#include <iostream>

#include "inventory.h"

using namespace std;

void main()

{

inventory item(25);

item.purchase(18);

cout << item.getNumberOnHand(); // Output: \_\_\_\_\_\_

if(!item.purchase(21))

item.purchase(10);

cout << item.getNumberOnHand(); // Output: \_\_\_\_\_\_

}

[14] The function slideBack() scans elements in an array starting at the first element. The function looks at successive pairs of elements and exchanges their values if the first element is greater than the second element. At the end of the scan, the largest element should appear at the back (end) of the array. We provide the following incorrect implementation for slideBack().

void slideBack(int arr[], int n)

{ int i = 0;

for (i = 0; i < n; i++)

if (arr[i] > arr[i+1])

swap(arr[i], arr[i+1]);

}

What is the error in the implementation of the function?

[15] In the employee class below, the member function setPayRate() throws an employeeError exception with the message "Below minimum wage". Assume the constant real number MINPAYRATE defines the current minimum wage.

class employee

{

public:

. . .

void setPayRate(double pay);

private:

. . .

double payRate;

};

(a) Complete the implementation of the function setPayRate()

void employee::setPayRate(double pay) { }

(b) Assume empObj is an object of type employee. Fill in the missing statements in a code sequence that calls the member function and outputs the error message if an exception occurs.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

{

empObj.setPayRate(6.50);

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

{

// Output the message

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

[16] The cylinder class provides measurement of a cylinder object, which we can view as a 3-dimensional figure created by a circle sliding vertical upward along a line denoting the height. A cylinder is defined by the radius of the circular base and the height. Using object composition, we represent the base using a circle object.

class cylinder {

public:

cylinder(double r, double h): \_\_\_\_\_\_\_, height(h)

{}

 double volume() // area of base \* height

{ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_}

double getRadius() // radius of the base

{ return \_\_\_\_\_\_\_\_ }

private:

circle base;

double height;

};

The following is the circle class implementation using inline code.

const double PI = 3.14159;

class circle

{

public:

circle(double r = 0.0): radius(r)

{}

double getRadius() const

{ return radius; }

void setRadius(double r)

{ radius = r; }

double area() const

{ return PI \* radius \* radius; }

double circumference() const

{ return 2.0 \* PI \* radius; }

private:

double radius;

};

(a) In the implementation of the cylinder constructor, complete the initialization list.

(b) Give the return value for getRadius().

(c) Implement the member function volume().

[17] The following is the declaration of the operator +, which is overloaded as a friend function in the time24 class.

friend time24 operator+ (const time24& lhs, const time24& rhs);

(a) Modify the declaration so that the operator becomes a member function of the class.

(b) Implement the member function using inline code.

[18] The class length stores the linear measure of an object with integer data members feet and inch.

class length

{

public:

// constructor initializes feet and inches

length(int ft = 0, int in = 0);

int getFeet() const; // return the feet value for the current length

int getInch() const; // return the inch value for the current length

double getLength() const; // return the length in units of feet

// compare the total length of the two operands

friend bool operator== (const length& lhs, const length& rhs);

friend bool operator< (const length& lhs, const length& rhs);

// form and return lhs + rhs; store in standard form

friend length operator+ (const length& lhs, const length& rhs);

// form and return lhs - rhs; store in standard form

// Precondition: lhs >= rhs. if not, throw rangeError

friend length operator- (const length& lhs, const length& rhs);

// current object = object + rhs; store in standard form

// Postcondition: the length increases by the value of rhs

length& operator+= (const length& rhs);

// output obj in the format ft'in"

friend ostream& operator<< (ostream& ostr, const length& obj);

private:

int feet;

int inch;

// utility function sets inches in range from 0 to 11

void standardLength();

};

(a) Implement operator+=.

(b) In the implementation of operator-, which of the following statements throws a rangeError exception.

(i) rangeError = exception("length: invalid subtraction");

(ii) throw rangeError = exception("length: invalid subtraction");

(iii) throw exception rangeError("length: invalid subtraction");

(iv) throw rangeError("length: invalid subtraction");

(c) Implement member function getLength().return feet + inch/12.0;

(d) Give the output of the (cout) statements in the following code.

length lenA(2,3), lenB(4,6), lenC(8,10), lenD;

cout << lenA.getLength(); // output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lenA += lenC;

cout << lenA.getFeet(); // output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lenD = lenB + lenC;

cout << lenD; // output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[19] Show the order of elements for intArr after the first two passes of the selection sort.

int intArr[] = {5, 10, 7, 2, 24, 19};

Pass 0: \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_

Pass 1: \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_

[20] Use the following array to trace the first two iterations of the binary search algorithm for the specified target value. For each iteration, indicate the midpoint index, and the index range [first, last) for the sublist in the next iteration.



(a) Target 7

Iteration 1: Midpoint index: \_\_\_\_\_\_\_ Index range for the next sublist: [ \_\_\_\_, \_\_\_\_ )

Iteration 2: Midpoint index: \_\_\_\_\_\_\_ Index range for the next sublist: [ \_\_\_\_, \_\_\_\_ )

(b) Target 55

Iteration 1: Midpoint index: \_\_\_\_\_\_\_ Index range for the next sublist: [ \_\_\_\_, \_\_\_ )

Iteration 2: Midpoint index: \_\_\_\_\_\_\_ Index range for the next sublist: [ \_\_\_\_, \_\_\_ )

[21] Assume T(n) is a count of the number of key operations for an algorithm that processes a list of n elements. Determine the Big-O runtime efficiency of each of the following algorithms:

(a) T(n) = n - 18 (b) T(n) = n3 + 6n + 5 (c) T(n) =



(d) T(n) = (e) T(n) = 4

[22] For each of the following functions T(n), select the appropriate Big-O value from the following list

(i) O(n) (ii) O(n2) (iii) O(n3) (iv) O(2n) (v) O(1)

(a) T(n) = 2n - 3 \_\_\_\_\_\_\_\_\_\_

(b) T(n) = (n3 + 3n + 7)/(n - 2) \_\_\_\_\_\_\_\_\_

(c) T(n) = (2n2 - 3)/(n2 + 1) \_\_\_\_\_\_\_\_\_

[23] The function shiftLeft() slides the elements in an array to the left one position and rotates the first element to the back of the array.

void shiftLeft(int arr[], int n)

{

int i, temp = arr[0];

for (i = 0; i < n-1; i++)

arr[i] = arr[i+1];

arr[n-1] = temp;

}

Write a template version of shiftRight().

[24] Trace the recursive function and give the output for the specified function calls

void f(char ch)

{

if (ch >= 'a' && ch <= 'h')

{

cout << ch;

f(ch + 1);

}

else

cout << endl;

}

(a) f('d') Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(b) f('a') Output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[25] Give the first seven terms in the sequence created by the following recursive function f().

int f(int n)

{

if (n <= 1)

return n;

else

return f(n-1) + 2\*f(n-2);

}

Sequence: \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_

[26] From the following list of containers, choose the one that is most appropriate for the problem.

(i) array (ii) vector (iii) list (iv) stack (v) queue

(a) Teller line in a bank.

(b) List of return addresses for a chain of recursive function calls.

(c) Sequence that efficiently maintains an ascending order when adding a new element.

(d) List of days in the month for the months January through December.

(e) Indexed list to store input from a file. Input terminates at end-of-file.

[27] A call to function f() assigns the return vector to resultV. List the contents of resultV.

template <typename T>

vector<T> f(const vector<T>& x, const vector<T>& y)

{ vector<T> v = x;

int i;

for (i=0;i < y.size();i++)

v.push\_back(y[i]);

return v;

}

int a[] = {4, 7, 5, 2, 3}, b[] = {15, 18, 25};

int aSize = sizeof(a)/sizeof(int), bSize = sizeof(b)/sizeof(int);

vector<int> va(a, a+aSize), vb(b, b+bSize), resultV;

resultV = f(va, vb);

resultV = \_\_\_\_

[28]The following code fragment uses the vector class. Answer questions (a) - (c).

template <class T>

T func(const vector<T>& v, int m, int n)

{

return v[m] + v[n];

}

void main()

{ vector<int> intV;

string strArr[] = {"ADT", "C++", "Class"};

int strArrSize = sizeof(strArr)/sizeof(string);

vector<string> strV(strArr,strArr+ strArrSize);

for(int i=0;i < 5;i++)

intV.push\_back(i+2); // <see part a>

cout << func(intV,0,3) << endl; // output 1: <see part b>

cout << func(strV,1,2) << endl; // output 2: <see part c>

}

(a) In the main program, vector intV holds integer values. Give the set of values that are assigned to the list. \_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**\_\_\_\_\_

(b) What is produced by output 1?

(i) 4 (ii) 7 (iii) 2 (iv) 6

(c) The value of output 2 is:

(i) ADT (ii) Class (iii) C++Class (iv) C++ADT

43.

[29] Show the values in each vector.

int intArr[] = {2, 5, 11, 9, 25};

vector<int> vA(3), vB(5,2), vC(intArr, intArr+5);

|  |  |  |  |
| --- | --- | --- | --- |
| vA |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| vB |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| vC |  |  |  |  |  |

[30] Implement the function diffVector() that takes a vector v and returns a new vector whose elements are the difference between successive elements in vector v. For instance, assume v has the elements {6, 2, 9, 5, 8, 10}. The diffVector(v) returns the vector with elements {-4, 7, -4, 3, 2}.

[31] Identify the error that occurs in each of the following statements.

(a) vector<int> \*vPtr;

vPtr = new vector<int>[15];

(b) int a[20], b[10];

b = a;

(c) int \*p = new int{1,4,9};

[32] Apply the function f() for parts (a) and (b).

template <typename T>

void f(T \*arr, int n)

{

T \*p = arr, \*q = arr+n/2, temp;

for(int i = 0; i < n/2; i++)

{

temp = \*p;

\*p++ = \*q;

\*q++ = temp;

}

}

(a) What are the contents of arr after calling f()?

int arr[] = {3, 5, 7, -9, 6};

f(arr, 5);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Contents |  |  |  |  |  |

(b) What are the contents of strArr after calling f()?

string strArr[] = {"North", "South", "East", "West"};

f(strArr, 4);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Contents |  |  |  |  |

[33] A sequence of statements use miniVector (done in the classroom) objects vA and vB. Assume the statements execute sequentially so that you must use the results from all previous statements to determine the size and the capacity of the miniVector object after executing the current statement. Assume default capacity is 2.

miniVector<int> vA(4); // vA.vSize = \_\_\_\_ vA.vCapacity = \_\_\_\_

miniVector<int> vB; // vB.vSize = \_\_\_\_ vB.vCapacity = \_\_\_\_

vA.push\_back(5); // vA.vSize = \_\_\_\_ vA.vCapacity = \_\_\_\_

vB.push\_back(5): // vB.vSize = \_\_\_\_ vB.vCapacity = \_\_\_

vA.push\_back(6); // vA.vSize = \_\_\_\_ vA.vCapacity = \_\_\_\_

vB.push\_back(6): // vB.vSize = \_\_\_\_ vB.vCapacity = \_\_\_\_

[34] Correct any syntax error in the declaration of the vector or miniVector object.

(a) vector<vector<int>> table

(b) vector<int, int> > table;

(c) miniVector<string> vString[15];

[35] The following is a 4 by 4 table of integer values with the row and column labels in bold.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** |
| **0** | 5 | 8 | 7 | -1 |
| **1** | 3 | 0 | 2 | 9 |
| **2** | -4 | 7 | 6 | 2 |
| **3** | 1 | 0 | 7 | 4 |

(a) Declare the table as the object mat of type matrix where matrix is a template class that has a constructor matrix(int rows, int cols).

(b) Assuming mat has the integer values above, and writeVector is a function that displays one matrix row. What is the output from writeVector(mat[1])? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

For Questions 36 and 37, add the member functions insert() and erase() to the miniVector class. For their implementation, use the following declaration for key members in the private section of the class.

class miniVector

{

private:

int vSize; // number of elements in the list

T \*vArr; // the dynamic array

}

[36] The function insert() adds the new element item at index pos in the vector.

void insert(int pos, const T& item);

Complete the implementation of insert() using inline code.

void insert(int pos, const T& item){

int i;

// shift elements on the tail of the list to the right

// one position

for

{

}

vArr[pos] = item;

}

[37] The function erase() removes from the vector the element at index pos.

void erase(int pos);

Complete the implementation of erase() using inline code.

void erase(int pos)

{

int i;

// shift elements on the tail of the list to fill the gap

for

{

}

}

[38] What is the output from the following sequence of instructions?

int arr[] = {5, 9, 2, 7};

int arrSize = sizeof(arr)/sizeof(int);

list<int> alist(arr, arr+arrSize);

alist.pop\_front();

cout << alist.back(); // Output: \_\_\_\_\_\_\_\_\_\_

cout << alist.front(); // Output: \_\_\_\_\_\_\_\_\_\_

alist.pop\_back();

cout << alist.back(); // Output: \_\_\_\_\_\_\_\_\_\_

alist.pop\_front();

alist.push\_front(7);

alist.back() = alist.front();

writeList(alist); // a function prints entire list

Output: \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_ \_\_\_\_\_\_

[39] Assume the declaration of array weekName.

string weekName[] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat");

const int DAYSINWEEK = 7;

(a) Declare the list object weekList to be a list of strings whose initial values are the names of the days of the week.

For parts (b) and (c), use the iterators strIterA and strIterB.

list<string>::iterator strIterA, strIterB;

(b) After executing the following instructions, strIterA is the location of day \_\_\_\_\_\_\_\_\_.

strIterA = weekList.end();

strIterA--;

strIterA--;

(c) After executing the loop, strIterA is the location of day \_\_\_\_\_\_\_\_\_\_\_\_

strIterA = weekList.begin();

strIterB = weekList.end();

strIterB--;

while(strIterA != strIterB)

{

strIterA++;

strIterB--;

}

[40] After executing the following statements, what is the resulting list?

list<string> alist;

list<string>::iterator iter;

alist.push\_front("tom");

alist.push\_front("sue");

iter = alist.begin();

iter++;

alist.insert(iter,"dave");

alist.insert(iter,"bill");

alist: \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

[41] Study the function below and give the output of the statements that follow.

template <typename T>

list<T> f(const list<T>& alist)

{ list<T> copyList = alist;

list<T>::iterator aIter = copyList.begin(), bIter;

while (aIter != copyList.end())

{ bIter = aIter;

bIter++;

while (bIter != copyList.end())

if (\*bIter == \*aIter)

{ copyList.erase(bIter++);

copyList.insert(aIter,\*aIter);

}

else

bIter++;

aIter++;

}

return copyList;

}

int arr[] = {5, 1, 2, 5, 3, 1, 5, 3, 9};

int arrSize = sizeof(arr)/sizeof(int);

list<int> intList(arr, arr+arrSize);

intList = f(intList);

Elements in intList: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[42] Consider the lists

list<int> intList, s, t;

Assume intList contains the elements {48, 7, 15, 26, 4, 31, 10, 12}. What are the contents of lists s and t after the function call

splitList(intList, s, t);

void splitList(const list<int>& aList, list<int>& one, list<int>& two)

{ list<int>::const\_iterator iter = aList.begin();

int n = 1;

while (iter != aList.end())

{ if (n%2 == 1)

one.push\_back(\*iter);

else

two.push\_back(\*iter);

iter++;

n++;

}

}

[43] Assume the following declarations:

list<int> intList;

list<int>::iterator intIter;

and that the list already has the elements 2 8 5

In (a)-(c), what are the contents of the list after each statement sequence? Note: The statements are cumulative.

(a) intIter = intList.begin();

intList.insert(intIter,4); list: \_\_\_\_

(b) intIter = intList.end();

intList.insert(intIter,15); list: \_\_\_\_

(c) intIter = intList.begin();

intIter++;

intIter++;

intList.erase(intIter++); list: \_\_\_\_

intList.insert(intIter,3); list: \_\_\_\_

[44] What is the output from the following sequence of stack operations?

stack<int> stk;

int x = 3, y = 19;

stk.push(6);

stk.push(x);

stk.push(y);

cout << stk.top(); // Output 1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

stk.pop();

stk.top() \*= 4;

stk.push(35);

while (!stk.empty())

{ y = stk.top();

cout << y << " "; // Output 2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

stk.pop();

}

[45] Assume alist is a list of integer elements and stk is a stack of integers. A pair of loops first copies the elements from the list to the stack and then copies the elements from stack back to the list. For each part, compare the order of the elements in the original list and the final list.

(a) Loop 1:

while(!alist.empty())

{

stk.push(alist.back());

alist.pop\_back();

}

Loop 2:

while(!stk.empty())

{

alist.push\_front(stk.top());

stk.pop();

}

(i) The two copies produce the same ordering in the new list

(ii) The two copies reverse the ordering in the new list

(b) Loop 1:

while(!alist.empty())

{

stk.push(alist.front());

alist.pop\_front();

}

Loop 2:

while(!stk.empty())

{

alist.push\_front(stk.top());

stk.pop();

}

(i) The two copies produce the same ordering in the new list

(ii) The two copies reverse the ordering in the new list

[46] In each part, give the postfix (RPN) expression for the corresponding infix expression

(a) Infix: a \* b + (c - d) Postfix: \_\_\_\_ \_\_\_\_\_\_

(a) Infix: (a + b \* c) /(d - e) Postfix: \_\_\_\_\_\_\_\_\_\_\_\_

[47] In each part, give an infix expression that represents the postfix (RPN) expression

(a) Postfix: a b c + d - \* Infix: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(b) Postfix: a b + c d - e \* / Infix: \_\_\_\_\_ \_\_\_\_\_\_

[48] A call to function f() reorders the elements in a list. For each part, give the order of the elements in the list after calling f().

template <typename T>

void f(list<T>& alist, const T& item)

{

stack<T> stk;

list<T>::iterator iter = alist.begin();

while (iter != alist.end())

if (\*iter < item)

{

stk.push(\*iter);

alist.erase(iter++);

}

else

iter++;

while (!stk.empty())

{

alist.push\_front(stk.top());

stk.pop();

}

}

(a) List intList initially has elements {7, 3, 14, 5, 12, 20, 6};

f(intList, 10);

List after call to f(): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(b) List charList initially has elements {'g', 't', 'a', 'd', 'p', 'f', 'k', 'w'};

f(charList, 'g');

List after call to f(): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[49] The following figure displays the System Stack after the "push" phase in evaluating the recursive function f().

int f(int n)

{

if (n > 0)

return n + f(n-1); // address RetLoc2

else

return 0;

}

int value = f(4); // calling statement at address RetLoc1



Fill-in the return value as the activation records are popped from the stack.

[50] The question explores the operator stack in the converting of an infix expression to a postfix (RPN) expression. After scanning an operator, a '(', or a ')', display the operator stack and the current operands in the RPN expression.

Infix expession: a/(b + (c - d) \* e)

(a) After scanning / (b) After scanning (

RPN Exp \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ RPN Exp \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Stack: Stack:

(c) After scanning + (d) After scanning (

RPN Exp \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ RPN Exp \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Stack: Stack:

(e) After scanning - (f) After scanning )

RPN Exp \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ RPN Exp \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Stack: Stack:

(g) After scanning \* (h) After scanning )

RPN Exp \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ RPN Exp \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Stack: Stack: