**CS121 Assignment #03 Unsorted List**

Use the linked list pictured below for Problems 1 through 4.

Diagram

Description automatically generated

1. Give the values of the following expressions:

**a.** ptr1 🡪 info

**b.** ptr2 🡪 next 🡪 info

**c.** listData 🡪 next 🡪 next 🡪info

1. Are the following expressions true or false?

**a.** listData 🡪 next == ptr1

**b.** ptr1 🡪 next 🡪 info == 60

**c.** ptr2 🡪 next 🡪 == NULL

**d.** listData 🡪 info == 25

1. Declare whether the syntax of each of the the following statements is valid or invalid. If it is valid, mark it OK; if it is invalid, explain what is wrong.

**a.** listData 🡪 next = ptr1 🡪 next;

**b.** listData 🡪 next = \*(ptr2 🡪 next);

**c.** \*listData = ptr2;

**d.** ptr2 = ptr1 🡪 next 🡪 info;

**e.** ptr1 🡪 info = ptr2 🡪 info;

**f.** ptr2 🡪 ptr2 🡪 next 🡪 next;

1. Write one statement to do each of the following:

**a.** Make listData point to the node containing 45.

**b.** Make ptr2 point to the last node in the list.

**c.** Make listData point to an empty list.

**d.** Set the info member of the node containing 45 to 60.

1. The following program has careless errors on several lines. Find and correct the errors and show the output where requested.

#include <iostream>

int main()

{

int\* ptr;

int\* temp;

int x;

ptr = new int;

\*ptr = 4;

\*temp = \*ptr;

cout << ptr << temp;

x = 9;

\*temp = x;

cout << \*ptr << \*temp;

ptr = new int;

ptr = 5;

cout << \*ptr << \*temp; // output: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

return 0;

}

Problem 6 through 14 refer to blanks in the following code segment.

Class UnsortedType

{

public:

//all the prototypes go here.

private:

int length;

NodeType\* listData;

};

void UnsortedType::DeleteItem(ItemType item)

// Pre: Item is in list

{

NodeType\* tempPtr; // pointer delete

NodeType\* predLoc; // trailing pointer

NodeType\* location; // traveling pointer

bool found = false;

location = listData;

predLoc = \_\_\_\_\_\_\_\_\_\_\_\_\_; // 6

length--;

// Find node to delete.

while (\_\_\_\_\_\_\_\_\_\_\_\_) // 7

{

switch (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) // 8

{

case GREATER: ;

case LESS : predLoc = location;

location = \_\_\_\_\_\_\_\_\_\_\_; // 9

break;

case EQUAL : found = \_\_\_\_\_\_\_\_\_\_\_; // 10

break;

}

}

// delete location

tempPtr = \_\_\_\_\_\_\_\_\_\_\_\_\_; // 11

if (\_\_\_\_\_\_\_\_\_\_\_\_) // 12

\_\_\_\_\_\_\_\_\_\_\_\_ = location->next; // 13

ele

predLoc->next = \_\_\_\_\_\_\_\_\_\_\_\_\_; // 14

delete tempPtr;

}

**6.** Read the code segment above and fill in blank # 6.

A. NULL

B. True

C. false

D. listData

E. answer not shown

**7.** Read the code segment above and fill in blank # 7.

A. true

B. !found

C. false

D. moreToSearch

E. answer not shown

**8.** Read the code segment above and fill in blank # 8.

A. item.ComparedTo(listData->info)

B. item.ComparedTo(location->next)

C. item.ComparedTo(location->info)

D. item.CompareedTo(location)

E. answer not shown

**9.** Read the code segment above and fill in blank # 9.

A. item

B. \*location.next

C. (\*location).next

D. predLoc

E. answer not shown

**10.** Read the code segment above and fill in blank # 10.

A. false

B. true

C. predLoc == NULL

D. location != NULL

E. answer not shown

**11.** Read the code segment above and fill in blank # 11.

A. preLoc

B. location

C. predLoc->next

D. location->next

E. answer not shown

**12.** Read the code segment above and fill in blank # 12.

A. predLoc == NULL

B. location == NULL

C. predLoc == location

D. predLoc->next == NULL

E. answer not shown

**13.** Read the code segment above and fill in blank # 13.

A. predLoc

B. location

C. location->next

D. listData

E. answer not shown

**14.** Read the code segment above and fill in blank # 14.

A. listData

B. predLoc->next

C. location->next

D. newNode->next

E. answer not shown