Worksheet 5

Task 1

1. A linked list is a linear data structure where each element
2. Stores only data.
3. Stores data and a reference to the previous element.
4. Stores data and a reference to the next element
5. Requires random access for efficient operations
6. In asymptotic notation, we compute
7. Time complexity.
8. Space complexity.
9. Running complexity.
10. None of these
11. The minimum size of a MyNode in a singly linked list depends on the system architecture. Which of the following does not affect the minimum size of a MyNode in a singly linked list?
12. Size of the data stored in the MyNode (e.g., integer, string)
13. Number of elements in the entire linked list
14. System architecture (32-bit vs 64-bit)
15. Presence of a pointer to the next MyNode
16. The best case time complexity for traversing a singly linked list is \_\_\_\_\_ .
17. O(1)
18. O(n2)
19. O(n)
20. None of the above
21. A stream is
22. a library function
23. a system call
24. a source or destination of data that may be associated with a disk or other I/O devices
25. a file
26. Identify the token pasting operator.
27. +
28. ++
29. #
30. ##
31. Which of the following is the correct function prototype for the function main()?
32. main(char argc, char \*argv)
33. main(int argc, int \*argv)
34. main (int argc, int \*\*argv [] )
35. main(int argc,char \*argv[])
36. Identify the valid data type of the variable fraction in the following code.

typedef float HOST;

HOST fraction;

1. int and HOST
2. struct and HOST
3. enum and HOST
4. float and HOST
5. It is required to insert a MyNode at the end of a singly connected linked list having n MyNodes.

How many MyNodes are to be traversed for this insertion?

1. 1
2. n/2
3. n
4. none of these
5. n elements of a queue are to be reversed using another queue. The number of add and remove operations required to do so is
6. 2 x n
7. 4 x n
8. n
9. The task cannot be accomplished.

Task 2a

Ans:

#include <iostream>

#include <fstream>

#include <cstring> // for strcmp

#include <limits> // Required for numeric\_limits

using namespace std;

// Define the Employee structure

struct Employee {

char name[150];

int taxNumber;

double salary;

};

int main() {

Employee employees[50]; // Array to hold up to 50 employees

int count = 0; // Counter for the number of employees entered

cout << "Enter employee data (enter 'fin' as name to finish):" << endl;

// Reading employee data

for (int i = 0; i < 50; ++i) {

cout << "Enter name: ";

cin.getline(employees[i].name, 150);

if (strcmp(employees[i].name, "fin") == 0) break; // Stop input if name is 'fin'

cout << "Enter tax number: ";

if (!(cin >> employees[i].taxNumber)) {

cout << "Invalid tax number, please enter a valid number." << endl;

cin.clear(); // Clear error flag

cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // Discard the rest of the line

--i; // Decrement i to retry the same index

continue; // Continue to the next iteration

}

cout << "Enter salary: ";

if (!(cin >> employees[i].salary)) {

cout << "Invalid salary, please enter a valid number." << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

--i;

continue;

}

cin.ignore(); // Ignore the newline character left in the input buffer

count++; // Increment the count of valid employees

}

// Write to a binary file

ofstream outFile("test.bin", ios::binary);

if (!outFile) {

cerr << "Error opening file for writing." << endl;

return 1; // Return error code 1

}

// Write each employee structure to the binary file

outFile.write(reinterpret\_cast<char\*>(employees), sizeof(Employee) \* count);

outFile.close(); // Close the file stream

cout << "Data written to test.bin successfully." << endl;

return 0; // Successful execution

}

Task 2b

Ans-

#include <iostream>

#include <fstream>

using namespace std;

// Define the Employee structure as used before

struct Employee {

char name[150];

int taxNumber;

double salary;

};

const int MAX\_EMPLOYEES = 50;

int main() {

Employee employees[MAX\_EMPLOYEES];

Employee filteredEmployees[MAX\_EMPLOYEES];

double thresholdSalary;

double totalSalary = 0;

int numEmployees = 0;

int filteredCount = 0;

// Read from test.bin

ifstream inFile("test.bin", ios::binary);

if (!inFile) {

cerr << "Could not open test.bin for reading." << endl;

return 1;

}

// Load employees from the file

while (inFile.read(reinterpret\_cast<char\*>(&employees[numEmployees]), sizeof(Employee))) {

if (++numEmployees == MAX\_EMPLOYEES) break;

}

inFile.close();

// Ask the user for the salary threshold

cout << "Enter the salary threshold: ";

cin >> thresholdSalary;

// Filter employees whose salary is more than the threshold and calculate total salary for average calculation

ofstream outFile("data.bin", ios::binary);

if (!outFile) {

cerr << "Could not open data.bin for writing." << endl;

return 1;

}

for (int i = 0; i < numEmployees; ++i) {

if (employees[i].salary > thresholdSalary) {

outFile.write(reinterpret\_cast<char\*>(&employees[i]), sizeof(Employee));

filteredEmployees[filteredCount++] = employees[i];

totalSalary += employees[i].salary;

}

}

outFile.close();

// Calculate the average salary if there are any employees above the threshold

if (filteredCount > 0) {

double averageSalary = totalSalary / filteredCount;

cout << "Average salary of employees stored in data.bin: " << averageSalary << endl;

} else {

cout << "No employees have a salary above the threshold." << endl;

}

return 0;

}

Task 3

Ans-

#include <iostream>

using namespace std;

// Define a structure for a node in the linked list

struct MyNode

{

int data;

struct MyNode \*next;

};

// Function to insert a new node at the beginning of the list

void push(struct MyNode\*\* head, int MyNode\_data)

{

/\* 1. Allocate memory for a new node \*/

struct MyNode\* newNode = new MyNode;

/\* 2. Set the data for the new node \*/

newNode->data = MyNode\_data;

/\* 3. Set the next pointer of the new node to point to the current head \*/

newNode->next = \*head;

/\* 4. Write a code to move the head to point to the new MyNode \*/

\*head = newNode;

}

// Function to insert new MyNode after a given node

void insertAfter(struct MyNode\* prev\_MyNode, int MyNode\_data)

{

/\* 1. check if the given prev\_MyNode is NULL \*/

if (prev\_MyNode == NULL)

{

cout << "Previous node cannot be NULL";

return;

}

/\* 2. create a new MyNode \*/

struct MyNode\* newNode = new MyNode;

/\* 3. Give data to the MyNode \*/

newNode->data = MyNode\_data;

/\* 4. Adjust pointers to insert the new node after the given node \*/

newNode->next = prev\_MyNode->next;

/\* 5. move the next of prev\_MyNode as new\_MyNode \*/

prev\_MyNode->next = newNode;

}

/\* Function to insert new node at the last of the linked list \*/

void insertNode(struct MyNode\*\* head, int MyNode\_data)

{

/\* 1. Allocate memory for a node \*/

struct MyNode\* newMyNode = new MyNode;

/\* 2. set data for the MyNode \*/

newMyNode->data = MyNode\_data;

/\* 3. set next pointer of new MyNode to null as its the last MyNode \*/

newMyNode->next = NULL;

/\* 4. if list is empty, new node becomes first node \*/

if (\*head == NULL)

{

\*head = newMyNode;

return;

}

struct MyNode \*last = \*head; /\* used in step 5 \*/

/\* 5. Otherwise go till the last node \*/

while (last->next != NULL) {

last = last->next;

}

/\* 6. Change the next of last node \*/

last->next = newMyNode;

}

// function to display linked list contents

void showList(struct MyNode \*node)

{

//Code to traverse the list to show each node

while (node != NULL)

{

cout << node->data << "-->";

node = node->next;

}

if (node == NULL)

cout << "null\n";

}

int main()

{

/\* empty list \*/

struct MyNode\* head = NULL;

// Insert 10.

insertNode(&head, 10);

// Insert 20 at the beginning.

push(&head, 20);

// Insert 30 at the beginning.

push(&head, 30);

// Insert 40 at the end.

insertNode(&head, 40);

// Insert 50, after 20.

MyNode\* temp = head; // Finding node with data 20

while (temp != NULL && temp->data != 20) {

temp = temp->next;

}

insertAfter(temp, 50);

cout << "Final linked list: " << endl;

showList(head);

return 0;

}