**Câu 1 (4 điểm): Giả sử người ta có nhu cầu dùng danh sách liên kết để lưu trữ các số nguyên dương. Anh/chị hãy thực hiện :**

**a. Định nghĩa cấu trúc dữ liệu để có thể lưu trữ như yêu cầu (1 điểm)  
b. Viết hàm nhập các số nguyên vào danh sách, việc nhập kết thúc khi nhập giá trị -1 (1 điểm)**

**c. Viết hàm void TinhToan(ptr Head) in ra giá trị lớn nhất, giá trị nhỏ nhất và giá trị trung bình của danh sách. Qui ước: nếu danh sách là rỗng thì các giá trị này đều là 0. (1 điểm)  
d. Viết hàm sắp xếp danh sách theo giá trị của phần tử trong danh sách giảm dần và in ra màn hình (1 điểm)**

**Giải**

**a.** Cấu trúc dữ liệu được định nghĩa như sau:

|  |
| --- |
| #include <iostream>  **using** **namespace** std;  **typedef** **struct** tagNode  {  **int** data;  **struct** tagNode\* Next;  } Node;  **typedef** **struct** tagList  {  Node\* Head;  Node\* Tail;  } List; |

**b.** Hàm nhập:

|  |
| --- |
| **void** InputList(List &l)  {  **int** x;  **do**  {  cin >> x;  **if** (x == -1)  **break**;  AddTail(l, CreateNode(x));  }  **while** (x != -1);  } |

**c.** Hàm tính toán theo yêu cầu:

Sơ lược về hàm:

- Ta khởi tạo các giá trị S, i, Min, Max, Avg và 1 Node p kiểu con trỏ để lưu giá trị l.Head của danh sách

- Trước tiên ta kiểm tra danh sách có rỗng không, nếu có thì xuất các giá trị 0 như yêu cầu đề, nếu không thì t duyệt qua danh sách.

- Mỗi lần duyệt thì biến i sẽ được tăng lên 1, để đếm số lượng phần tử có trong danh sách, đồng thời ta cũng tìm Max và Min sau mỗi lần duyệt.

- Sau khi duyệt xong, ta tiến hành tính giá trị trung bình của danh sách.

- Cuối cùng, xuất các giá trị theo yếu cầu của đề bài.

|  |
| --- |
| **void** TinhToan(List l)  {  **int** S = 0, i = 0;  **int** Min = l.Tail->data, Max = l.Head->data;  **float** Avg = 0.0;  Node\* p = l.Head;  **if** (IsEmpty(l) == **true**)  {  cout << "- The list is empty!\n";  S = 0; Min = 0; Max = 0; Avg = 0.0;  }  **else**  {  **while** (p != **NULL**)  {  S = S + p->data;  **if** (p->data > Max)  Max = p->data;  **if** (p->data < Min)  Min = p->data;  i++;  p = p->Next;  }  Avg = **float**(S/i);  }  cout << "- Tong cac phan tu trong danh sach: " << S << endl;  cout << "- Phan tu lon nhat trong danh sach: " << Max << endl;  cout << "- Phan tu nho nhat trong danh sach: " << Min << endl;  cout << "- Gia tri trung binh cac phan tu trong danh sach: " << Avg << endl;  }  /\*  Giải thích:  - S: là tổng các giá trị trong List  - i: là tổng số giá trị có trong List  - Max: là giá trị lớn nhất List  - Min: Là giá trị nhỏ nhất List  - Avg: là giá trị trung bình của List  \*/ |

**d.** Hàm sắp xếp (Quick Sort): Ta viết thêm 1 hàm PrinList, sau đó là hàm QuickSort. Ta xuống hàm main() để thực hiện gọi hàm QuickSort và in danh sách ra màn hình.

|  |  |
| --- | --- |
| **void** PrintList(List l)  {  Node \*p = l.Head;  **while** (p != **NULL**)  {  cout << p->data << " ";  p = p->Next;  }  }  **void** QuickSort(List &L)  {  Node \*p, \*X;  List L1, L2;    **if** (L.Head == L.Tail)  **return**;    CreateList(L1);  CreateList(L2);    X = L.Head;  L.Head = X->Next;    **while** (L.Head != **NULL**)  {  p = L.Head;  L.Head = p->Next;  p->Next = **NULL**;  **if** (p->data > X->data)  AddTail(L1, p);  **else**  AddTail(L2, p);  } | QuickSort(L1);  QuickSort(L2);    **if** (L1.Head != **NULL**)  {  L.Head = L1.Head;  L1.Tail->Next = X;  }  **else**  L.Head = X;    X->Next = L2.Head;    **if** (L2.Head != **NULL**)  L.Tail = L2.Tail;  **else**  L.Tail = X;  }  **int** main()  {  List L;  CreateList(L);  InputList(L);  TinhToan(L);  QuickSort(L);  cout << "- List: ";  PrintList(L);  } |