حل التكليف الأول نظري:

١. القائمة المرتبطة الأحادية(Singly Linked List)

تتكون من عقد تحتوي على بيانات ومؤشر يشير إلى العقدة التالية فقط.

الاستخدامات:

- تُستخدم في بناء هياكل البيانات الأخرى مثل المكدس.(Stack)
- إدارة الذاكرة البسيطة حيث يكون التصفح في اتجاه واحد كافياً.
- التطبيقات التي لا تتطلب الرجوع للخلف لتوفير الذاكرة

المميزات:

- تستهلك ذاكرة أقل مقارنة بالأنواع الأخرى لأن كل عقدة تخزن مؤشراً واحداً فقط.
- عمليات الإضافة والحذف في البداية سريعة جداً.

العيوب :

- لا يمكن التصفح أو الوصول إلى العناصر إلا في اتجاه واحد (من البداية للنهاية).
- صعوبة الوصول إلى العقدة السابقة لعقدة معينة دون البدء من رأس القائمة

2. القائمة المرتبطة المزدوجة(Doubly Linked List)

كل عقدة تحتوي على مؤشرين: واحد يشير للعقدة التالية والآخر يشير للعقدة السابقة

الاستخدامات:

- متصفحات الويب (للرجوع للخلف وللأمام في سجل الزيارات).
- خاصية "التراجع (Undo) "و "الإعادة (Redo) "في البرامج.
- خوارزميات الجدولة في أنظمة التشغيل.

المميزات:

- يمكن تصفح القائمة في كلا الاتجاهين (للأمام وللخلف).
- عملية حذف عقدة معينة أسهل لأننا نملك وصولاً مباشراً للعقدة السابقة

العيوب :

- تستهلك ذاكرة أكبر بسبب تخزين مؤشر إضافي (Previous pointer) لكل عقدة.
- العمليات البرمجية (الإضافة والحذف) أكثر تعقيداً قليلاً لأنها تتطلب تعديل مؤشرين

3. القائمة المرتبطة الدائرية(Circular Linked List)

هي قائمة (أحادية أو مزدوجة) حيث تشير العقدة الأخيرة فيها إلى العقدة الأولى، مما يشكل حلقة مغلقة

الاستخدامات:

- أنظمة التشغيل لجدولة المهام التي تعمل بنظام "التناوب.(Round Robin) "
- الألعاب التي تتطلب تبادل الأدوار بين اللاعبين بشكل مستمر.
- تطبيقات الوسائط المتعددة (مثل قوائم تشغيل الموسيقى التي تبدأ من جديد تلقائياً).

المميزات:

- يمكن الوصول إلى أي عقدة من أي نقطة بداية في القائمة.
- مثالية للتطبيقات التي تتطلب دوراناً مستمراً حول البيانات.

العيوب :

- أكثر تعقيداً في التنفيذ البرمجي لتجنب الدخول في حلقة لا نهائية (Infinite Loop) أثناء التصفح.
- تحديد نهاية القائمة يتطلب منطقاً برمجياً دقيقاً.