سوال اول :

الف)

1- درست

2- غلط. درمپ نباید کلیدهای مشابه وجود داشته باشد. اگر یک کلید را دوبار به مپ اضافه کنیم کلید و مقدار جدید جایگزین قبلی میشود.

ب)

1) primitive

2) HashSet

3) anonymous

4) ArrayList

5) استاتیک

ج)

1) اولین و مهم‌ترین تفاوت بین این دو در نحوه‌ی پیاده‌سازی این دو است. ArrayList در واقع یک آرایه‌ی داینامیک است، به این معنی که لازم نیست که ما از قبل سایز آن را مشخص کنیم و خود آن با هر بار اضافه و حذف داده آن را تغییر می‌دهد. این مانند وقتی است که ما در زبان C از malloc و calloc برای گرفتن آرایه در هیپ استفاده می‌کردیم. ولی پیاده‌سازی LinkedList در واقع به کل همان LinkedList دوطرفه است. یعنی node ها در جاهای مختلف حافظه قرار دارند و از طریق آدرس‌هایشان به یکدیگر متصل هستند. دستکاری کردن (حذف و اضافه کردن داده) در لینکدلیست سریع‌تر از اری‌لیست است و دلیل واضح آن همان نحوه‌ی پیاده‌سازی آن‌ها است. در اری‌لیست اگر بخواهیم داده‌ای را از میان داده‌ها حذف کنیم، در اکثرموارد نیاز است که بقیه را نیز شیفت بدهیم و این باعث کند شدن این کار می‌شود. در حالیکه در لینکدلیست خبری از شیفت دادن نیست و فقط با آدرس‌ها کار می‌شود. از تفاوت‌های دیگر این دو می‌توان به این اشاره کرد که اری‌لیست صرفا یک لیست است و حالت صفی ندارد ولی لینکدلیست هم یک لیست است و هم می‌تواند مانند صفی از داده‌ها رفتار کند. در کل می‌توان گفت که اری‌لیست برای ذخیره‌ی داده‌هایی که در آینده نیازی به دستکاری ندارند مناسب‌تر است ولی اگر قرار است بعدا تغییراتی در لیستمان به وجود بیاوریم، بهتر است از لینکدلیست استفاده کنیم.
2) در جاوا و بسیاری از زبان‌های دیگر، ما می‌توانیم تابعی با یک نام ولی متفاوت در ورودی داشته باشیم. به این کار overloading می‌گوییم. این تفاوت در ورودی می‌تواند به شکل‌های تفاوت در نوع ورودی‌ها، تفاوت در تعداد ورودی‌ها و یا تفاوت در هر دو باشد. این متدها می‌توانند در نوع خروجی با یکدیگر متفاوت باشند و تنها شرط ما کامل یکسان نبودن شکل ورودی آن‌ها است.
3) حلقه‌ی for-each حلقه‌ای است که بر مبنای پیمایش روی اعضای یک کالکشن بر مبنای افرایش تصاعدی index های آنان کار می‌کند. با حذف یک شی از کالکشن، در واقع شماره‌گذاری index برای اشیای بعد از آن شی تغییر می‌کند و این موجب این می‌شود که حلقه برای ادامه‌ی پیمایش بر روی کالکشن به مشکل بخورد. راه برطرف کردن این مشکل استفاده از iteraor به جای for-each است.
4) یک روش پیمایش بر روی key های این هش‌مپ است. این کار را می‌توان با استفاده از یک حلقه‌ی for-each انجام داد که فرم کلی آن به صورت for (keyType key : HashMapName.keySet()) انجام داد. همچنین برای پیمایش بر روی valueهای یک هش‌مپ نیز راه‌حل مشابهی وجود دارد که آن را می‌توان به صورت for (valueType value : HashMapName.values()) نمایش داد. اگر بخواهیم بر روی هر دو داده‌ای که هش‌مپ دارد به صورت همزمان پیمایش کنیم نیز راه‌حلی وجود دارد و آن استفاده از Map.Entry است. میتوان این کار را به صورت for (Map.Entry entry :

HashMapName.entrySet()) نشان داد و در بدنه‌ی حلقه با دستورات getValue() و getKey() میتوان به مقدار و کلید آن دسترسی داشت.

سوال دوم:

الف) یکی از نکات مهم گفته شده در سوال این است که نیازی به جست‌وجو در میان کاربران نداریم. همچنین مشخص است که لیست کاربران آنلاین در هر لحظه می‌تواند تغییر کند و کم یا زیاد شود. به همین دلیل ما باید دنبال کالکشنی باشیم که این کار باعث کاهش سرعت آن نشود و همچنین با توجه به اینکه زمان آنلاین‌شدن آن‌ها و ترتیبشان نیز مهم است میتوان فهمید که بهترین کالکشن برای ما IinkedList است زیرا سرعت برای ایجاد تغییر در آن مناسب است و داده‌ها دارای ترتیب هستند.

ب) در این لیست احتمال زیاد ما نیازی به ایجاد تغییر نداریم و تنها می‌خواهیم داده‌ها را در جایی ذخیره کنیم. همچنین نکته‌ی دیگر این است که این لیست اولا لازم نیست که ترتیب خاصی داشته باشد و ثانیا اسم هر نفر حداکثر باید یکبار در این لیست موجود باشد. با توجه به این نکات بهترین کالکشن برای نگه‌داری آن‌ها استفاده ازHashSet است که مجموعه‌ای از داده‌های یونیک و غیرتکراری است.

ج) برخلاف قسمت ه دراین قسمت روی سرعت جستجو در میان داده‌ها تاکید نشده است. همچنین با توجه به اینکه هدف ما از این کار صرفا ذخیره کردن داده‌ها و نه تغییر مداوم و چندباره‌ی آن‌ها است، کلاس ArrayList می‌تواند راه خوبی برای ذخیره‌ی آن‌ها باشد.

د) با توجه به تاکید سوال براینکه 1- حافظه محدود است و 2- امکان اینکه دانشجو به کلاس اضافه شود وجود ندارد، گزینه‌ی خوبی برای پیاده‌سازی کلاس می‌تواند استفاده از آرایه‌ای از دانش‌آموزان باشد. در آرایه امکان افزایش سایز وجود ندارد.

ه) با توجه به این که یک مورد مهم در اینجا جست‌وجوی سریع میان دانشجویان است میتوان گفت کالکشن مناسب می‌تواند استفاده از هش‌مپ باشد. به این شکل که ما شماره دانشجویی هر دانشجو که در واقع باید برای هر دانشجو متفاوت باشد را در قسمت key هش‌مپ قرار دهیم و اطلاعات دانشجو که خود به صورت یک آبجکت از یک کلاس دارای فیلدها و متودهای مختلف است را در قسمت valueهای این هش‌مپ قرار دهیم.