باسمه تعالی

دوره‌ی آموزشی المپیاد کامپیوتر

آزمون پایان دوره‌ای درس زبان

یک‌شنبه ۴ شهریور ۱۳۸۶

اویس قرن — وقت: ۱۰۰ دقیقه

نکات:

- جمع نمره‌ی سوالهای این امتحان ۱۰۰ نمره است.
- در این امتحان شما فقط حق دارید از مطالبی که در کلاس به شما درس داده شده استفاده کنید.
- شما **تنها** در سوالاتی که با ستاره مشخص شده‌اند می‌توانید از کتابخانه‌های زبان C یا C++ استفاده کنید.
- سوالاتی که از شما خواسته شده برنامه بنویسید، شما بایستی یک برنامه‌ی کامل شامل main، ورودی، خروجی و ...بنویسید. در سایر سوالات شما تنها بایستی چند تابع بنویسید.

مسئله‌ی اوّل: ..... ۱۰ نمره

مزایای استفاده از توابع در برنامه‌ی زبان C را بیان کنید به عبارت بهتر چرا همه‌ی توابع را در main اصلی ننویسیم؟؟؟ (۴ مورد)

مسئله‌ی دوم: ..... ۲۰ نمره

⋆⋆ در این سوال شما بایستی برنامه‌ای بنویسید که مشخصات یک گراف **بی‌جهت** نه لزوماً ساده (یعنی ممکن است بین برخی از رئوس بیش از یک یال باشد و یا ممکن است یک راس به خودش چندین یال داشته باشد) را از ورودی استاندارد بخواند.

در سطر اوّل فایل ورودی به ترتیب n و e تعداد راس‌ها و یال‌های گراف نوشته شده است. در e سطر بعدی در هر سطر دو عدد x و y نوشته شده که دو سر یکی از یالهای گراف را مشخص می‌کند.

فایل خروجی استاندارد بایستی n سطر داشته باشد که در سطر i ام همسایه‌های راس i نوشته شده‌اند. به ازای هر همسایه x [1] از راس i شما بایستی ابتدا x و سپس تعداد یالهایی که بین i و x وجود دارد را بنویسید.

مثال

```
Standard Input
2 4
1 1
1 2
```

[1] دقت کنید x می‌تواند مساوی i باشد

```
1 1
2 1

Standard Output
2 2 1 2
1 2
```

برنامه‌ی شما بایستی از $O(e \times \log n)$ باشد.
دقّت کنید مقدار ماکسیمم n مشخّص نیست. در ضمن اگر این سوال را با فرض $n < ۱۰۰۰۰$ حل کنید تنها ۵ نمره از شما کثر خواهد شد.

مسئله‌ی سوم: ..... ۲۰ نمره

در این سوال شما بایستی کاری شبیه به C++ در تخصیص حافظه و آزاد کردن حافظه از Heap انجام دهید. فرض کنید آرایه‌ی a از نوع int که بسیار بزرگ است وجود دارد. در این سوال شما بایستی توابع زیر را پیاده سازی کنید: **دقت کنید همه‌ی توابع باید از O(1) باشند.**

void initialize(): در این تابع شما می‌توانید متغیّرهایتان را برای مراحل بعدی مقدار دهی اوّلیّه بکنید. این تابع فقط یک‌بار قبل از استفاده از توابع بعدی اجرا خواهد شد.

int *allocateMemory(int min, int max): این تابع یک آرایه که عناصرش در بازه‌ی $[min, max)$ قرار دارد را در نظر می‌گیرد و اشاره‌گر به آن را بر می‌گرداند. دقّت کنید min یا max می‌توانند منفی هم باشند ولی می‌دانیم همیشه $min \leq max$ است. دقّت کنید عنصر max جز آن نیست. یعنی اگر $min = max$ باشد آرایه هیچ خانه‌ای ندارد.

void freeMemory(int *b): این تابع آرایه‌ای که قبلاً اختصاص داده شده بود و اشاره‌گر b به آن برگردانده شده بود را خالی می‌کند.

بعنوان یک مثال از نحوه‌ی استفاده از این توابع می‌توان نوشت:

```
intialize();
int *c=allocateMemory(-3, -2);
int *b=allocateMemory(0, n);
c[-3]=5;
for (int i=0; i<n; i++)
     b[i]=i;
freeMemory(b);
int *d=allocateMemory(-3, -3);
freeMemory(d);
freeMemory(c);
```

فرض کنید که آرایه‌ها به عکس ترتیبی که ساخته می‌شوند خالی می‌شوند یعنی اگر آرایه‌ی c زودتر از b ساخته شده باشد، b زودتر از c از بین می‌رود. شما برای پیاده‌سازی توابع فوق به غیر از آرایه‌ی a تنها می‌توانید از $O(۱)$ حافظه‌ی اضافی استفاده کنید. در ضمن در هر لحظه اگر جمعاً m تا آرایه allocate شده وجود داشته باشد که این آرایه‌ها جمعاً n خانه‌ی allocate شده دارند شما می‌توانید از $O(m+n)$ تا از خانه‌های آرایه‌ی a استفاده کرده باشید.

مسئله‌ی چهارم: ..... ۲۰ نمره

فرض کنید که داده‌ساختار لینک لیست **دو طرفه** از عناصری با type زیر ساخته شده است:

```
struct Node{
    Node *next, *prev;
    int key;
};
```

فرض کنید که next آخرین عنصر لینک لیست و prev اولین عنصر لینک لیست NULL است. شما بایستی تابع Node * insertLinkedList(Node *start1, Node *start2, Node *p, int where) را پیاده سازی کنید. start1 و start2 به ابتدای دو لینک لیست مختلف اشاره می‌کنند. این تابع باید لینک لیستی که start2 به آن اشاره می‌کند را در لینک لیستی که start1 به ابتدای آن اشاره می‌کند وارد کند. و اشاره‌گر به ابتدای لینک لیست حاصل را بعنوان خروجی برگرداند. p یکی از عناصر لینک لیستی است که start1 به آن اشاره می‌کند. در صورتی که where=0 باشد، لینک لیست دوم بایستی قبل از عنصر p وارد شود و اگر where=1 باشد، لینک لیست دوم باید بعد از عنصر p وارد شود. **دقت کنید شما باید این تابع را در $O(۱)$ پیاده سازی کنید**

بعنوان مثال اگر لینک لیست اوّل (از راست به چپ) شامل عناصر ۱ و ۲ و ۳ و ۴ باشد و p به عنصر اوّل (یعنی ۱) اشاره کند و لینک لیست دوم شامل عناصر ۵ و ٦ و ۷ و ۸ باشد، اگر insertLinkedList(start1, start2, p, 0) را اجرا کنیم لینک لیست اول به صورت ۵ و ٦ و ۷ و ۸ و ۱ و ۲ و ۳ و ۴ در می آید ولی اگر insertLinkedList(start1, start2, p, 1) را اجرا کنیم لینک لیست اول به صورت ۱ و ۵ و ٦ و ۷ و ۸ و ۲ و ۳ و ۴ در می آید.

مسئله‌ی پنجم: ..... ۳۰ نمره

در این سوال شما بایستی توابع lower_bound و upper_bound را پیاده سازی کنید. فرض کنید قرار است شما این توابع را روی آرایه‌ی **مرتب شده**[۲] از عناصری از یک تایپ به نام C اجرا کنید. در ضمن فرض کنید تابع bool f(C a, C b) وجود دارد که در صورتی که $a < b$، true و در غیر این‌صورت false برمی‌گرداند. شما بایستی تابع:

C *lower_bound(C *first, C *last, C key)

[۲] sort شده

را بنویسید. این تابع مشابه تابع lower_bound عادی در بازه‌ی first تا last به دنبال key می‌گردد. برای مقایسه‌ی عناصر از تابع f استفاده می‌کند و اشاره‌گر به عنصری که پیدا می‌کند را بعنوان خروجی برمی‌گرداند. در ضمن اگر بین first و last، n عنصر از تایپ C وجود داشته باشد این تابع باید در زمان $O(\log n)$ اجرا شود. به طور مشابه تابع C *upper_bound(C *first, C *last, C key) را بنویسید.

راهنمایی: اگر f(a,b)، false باشد و f(b,a) هم false باشد می‌توان نتیجه گرفت a=b است.

نکته: اگر نمی‌توانید سوال بالا را حل کنید:

همین توابع را برای int lower_bound(int l, int r, int key) و int uppper_bound(int l, int r, int key) بنویسید منتهی با این تفاوت که این توابع در آرایه‌ی **مرتّب شده** a که global و از نوع int است به دنبال key می‌گردند و اندیس خانه‌ای که پیدا می‌کنند را بعنوان خروجی برمی‌گردانند. در این صورت تنها ۱۵ نمره از این سوال را از دست خواهید داد.

موفق باشد