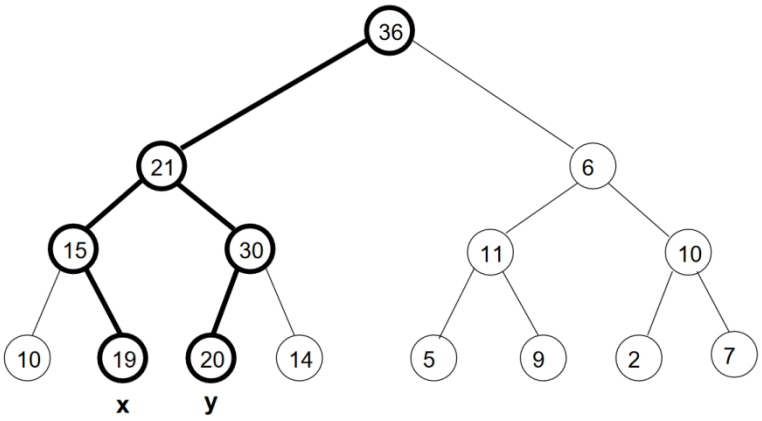
به نام او	**درس:** **ساختمان داده‌ها**	آذر ۱۴۰۰	دانشگاه علم و صنعت ایران
نام و نام خانوادگی: شماره دانشجویی:			

		نمره

۱	بدترین پیچیدگی زمانی شبه کدهای زیر را بنویسید:
	
	1) void F1 (int n, int sum) {
	for (int i = 0; i < n * 100; ++i) {
	for (int j = n; j > 0; j--)
	sum++;
	for (int k = 0; k < i; ++k)
	sum++;
	}
	}
	
	2) void F2 (int n, int sum) {
	int j = n;
	while (j > 2) {
	sum++;
	j = j / 2;
	}
	}
	
	3) int F3 (int n) {
	if (n < 10) {
	System.out.println ("!");
	return n+3;
	}
	else {
	return F3 (n-1) + F3 (n-1);
	}
	}
		۱.۵

۲	بدترین زمان اجرای $T(n)$ را برای توابع زیر بنویسید. 1) $T(n) = n^2\ lg^n + 2^n$ 2) $T(n) = n^2.\ (n + n \lg n + 1000)$ 3) $T(n) = n! + 2^{2n}$ 4) $T(n) = 1+1/2+1/3+\cdots+1/n$ 5) $T(n) = (\left(\frac{1}{2}\right)(3n + 5 + n))^4$ 6) $T(n) = T(n-2) + \lg n$	۳
۳	به سوالات زیر پاسخ دهید: ۱) اگر $f(n) = O(g(n))$ و $f(n) = \Omega(g(n))$ آنگاه $f(n)$ برابر است با: ۲) بدترین حالت اجرای $T(n)$ برای درج n عنصر در یک هیپ خالی (initially-empty heap) برابر است با: ۳) بدترین زمان اجرای $T(n)$ برای خروج یک عنصر از یک هیپ دودویی (binary heap) به اندازه n برابر است با: ۴) بدترین زمان اجرای $T(n)$ برای درج n عنصر در درخت جستجوی دودویی (BST) خالی برابر است با:	۲.۵
۴	درخت جستجوی دودویی (BST) را با درج اعداد زیر (با ۷۰ شروع و به ۶۲ ختم می‌شوند)، رسم کنید. 70 11 47 81 20 61 10 12 13 62 ۱) برای درخت بالا، گره‌ها را در یک پیمایش preorder بنویسید. ۲) برای درخت بالا، گره‌ها را در یک پیمایش postorder بنویسید. ۳) برای درخت بالا، گره‌ها را در یک پیمایش inorder بنویسید. در مورد پیمایش inorder درخت جستجوی دودویی چه چیزی متوجه شدید؟ ۴) پس از حذف مقدار ۱۱، درخت را دوباره ترسیم کنید.	۲
۵	فرض کنید یک هیپ کمینه‌ی دودویی (binary min-heap) با دقیقاً ۴ گره وجود دارد که حاوی موارد زیر به ترتیب زیر است (با ۳ شروع و به ۱۵ ختم می‌شوند): 3, 9, 11, 15 ۱) هر هیپ کمینه‌ی دودویی (binary min-heap) ممکن که می‌تواند با این توصیف مطابقت	۲

	داشته باشد را نشان دهید. برای هر کدام، درخت مناسب و نمایش آرایه را رسم کنید. ۲) برای یکی از پاسخ‌های بخش (۱)، نشان دهید که با ۴ عملیات deleteMin چه اتفاقی می‌افتد. به وضوح مشخص کنید که از کدام هیپ شروع می‌کنید و پس از هر deleteMin، هیپ را نشان دهید (شما می‌توانید فقط درخت را بعد از هر مرحله بکشید و نیازی به آرایه نیست).	
۶	شما یک درخت جستجوی دودویی (BST) با n عنصر دارید که دارای ارتفاع $h = O(\log(n))$ است، و باید k-امین بزرگ‌ترین عنصر درخت را پیدا کنید. آیا می‌توان k-امین بزرگ‌ترین عنصر را بدون اسکن تمام n عنصر (با فرض $k < n$) پیدا کرد؟ اگر بله، یک الگوریتم را بذکر پیچیدگی زمانی توصیف کنید . اگر نه، یک مثال نقض ارائه دهید.	۱.۵
۷	با کاوش خطی (linear probing)، شی در کدام ورودی جدول درهم‌سازی (hash table) سمت چپ با کلید ۱۵ ذخیره می‌شود؟ کدام ورودی برای کاوش درجه دوم (quadratic probing) استفاده می‌شود؟ اندازه جدول ۱۳ است و تابع درهم‌سازی (hash function) یک باقی‌مانده‌ی ساده از کلید عدد صحیح (integer key) است. در ادامه شاخص‌هایی که ناموفق بودند را بنویسید. **Index: / Key:** 0: 39 1: 53 2: 67 3: 81 4: 95 5: 18 6: 7: 33 8: 47 9: 61 10: 11: 12: 25	۱.۵
۸	فرض کنید جدول درهم‌سازی (hash table) زیر را دارید که با استفاده از کاوش خطی (linear probing) پیاده‌سازی شده است. تابع درهم‌سازی تابع درهم‌سازی (hash function) که ما از آن استفاده می‌کنیم، تابع $h(x) = x$ است. 0 1 2 3 4 5 6 7 8 9 18 12 3 14 4 21 ۱) به کدام ترتیب می‌توان عناصر را به جدول درهم‌سازی (hash table) اضافه کرد؟ (بیش از یک پاسخ صحیح وجود دارد و شما باید همه‌ی آن‌ها را بنویسید. فرض کنید اندازه‌ی جدول درهم‌سازی (hash table) هرگز تغییر نکرده است و هیچ عنصری هنوز حذف نشده است.) ۲) ۳ را از جدول درهم‌سازی (hash table) حذف کنید و جدول را رسم کنید.	۱.۵

۹	فرض کنید یک درخت دودویی کامل (complete binary tree) به ارتفاع h با $n = 2^h$ برگ به شما داده می‌شود، که در آن هر گره و هر برگ از این درخت یک "مقدار" مرتبط با v (یک عدد واقعی دلخواه) دارد. اگر x یک برگ باشد، مجموعه اجداد x را با A (x) نشان می‌دهیم (x به عنوان یکی از اجداد خود در نظر گرفته می‌شود). یعنی A (x) تا ریشه درخت از x، والد x، پدربزرگ و مادربزرگ و غیره تشکیل شده است. به همین ترتیب، اگر x و y برگ‌های متمایز باشند، اجداد x یا y را با A (x, y) نشان می‌دهیم. به این معنا که، $$A(x, y) = A(x) \cup A(y)$$ ۱) تابع f (x, y) را به عنوان مجموع مقادیر گره‌های A (x, y) تعریف کنید.  A (x, y) برجسته شده است. $$f(x, y) = 19+15+21+36+20+30 = 141$$ ۲) الگوریتمی ارائه دهید (شبه کد لازم نیست) که به طور موثر دو برگ x_0 و y_0 را پیدا کند. به طوری که $f(x_0, y_0)$ تا حد امکان بزرگ باشد. زمان اجرای الگوریتم شما چقدر است؟	۳

با آرزوی بهترین روزگار

ح.رحمانی