دانشکده مهندسی برق و کامپیوتر
مبانی کامپیوتر و برنامه‌سازی / تمرینات سری دهم / دی (۸۹-۱۳۸۸)
استاد احمدی فر کمک‌تدریس عباس نادی (abiusx@etebaran.com)

تمرین سری دهم ـ آمادگی برای امتحان پایانی (به همراه پاسخ)

تمرینات این سری کاملا اختیاری هستند :

۱. (متوسط) تابع swap زیر برای جابجایی هرگونه ساختار و متغیر ابتدایی نوشته شده است. ابتدا مشکل آنرا بفرمایید و سپس طوری آنرا اصلاح کنید که برای جابجایی هرگونه متغیر ساده یا مرکب کارا باشد:

```
template <class T>
void swap ( T & a, T & b)
{
  T  c=a;
  a=b;
  b=c;
}
```

دقت کنید که رشته‌های منتهی به صفر (char *) یک نوع ساده یا مرکب متغیر نیست بلکه یک استاندارد است بنابراین این تابع نباید اینگونه داده را منتقل کند.
راهنمایی : از دستورات memcpy و sizeof بهره بجویید.
مشکل آنست که عملگر مساوی (=) برای انواع مرکب (struct) کار نمی‌کند. بنابراین برای این دسته از انواع باید کار دیگری کنیم. از آنجایی که نمی‌دانیم یک نوع مرکب چه فیلدهایی دارد که آنها را مساوی قرار دهیم (در حالت کلی) در سطحی پایینتر کل حافظه آنرا منتقل می‌کنیم:

```
template <class T>
void swap ( T & a, T & b)
{
  T c;
  memcpy(&c , &a , sizeof(T));
  memcpy(&a , &b , sizeof(T));
  memcpy(&b , &c , sizeof(T));
}
```

۲. (آسان) برنامه‌ای بنویسید که دو عدد صد رقمی از ورودی گرفته و حاصل جمع آنها را خروجی دهد (جمع دو عدد صد رقمی حداکثر صد و یک رقمیست) اعداد مثبت و صحیح.
از آنجایی که اعداد در هیچ متغیر محاسباتی (عددی) جای نمی‌گیرند، و ورودی یک رشته ۱۰۰ رقمیست، آنرا به صورت رشته ورودی می‌گیریم و رقم به رقم جمع می‌کنیم:

```
char a[101],b[101] , s[102];
```

به خاطر اینکه یک کاراکتر نیز برای صفر آخر رشته کنار می‌گذاریم.

```
 cin >>a >>b ; //or gets them
for (int i = 0 ; i <100; ++i)
  s[ i + 1] = a[i] – '0' + b[i] – '0' ;
```

صفرها را به این دلیل کم کردیم که ارقام داخل رشته به صورت کاراکتر هستند نه عدد. بنابراین برای اینکه

کاراکتر ۷ به عدد ۷ تبدیل شود، باید آنرا از کاراکتر صفر کم کرد. در مرحله آخر کافیست Carry های تولید شده (یعنی جاهایی که حاصل جمع دو رقم بیشتر از ۱۰ شده) نرمالسازی کنیم:

```
s[0]=0; //last carry
for (int i=0; i<100; ++i)
  if (s[i+1] >= 10) { s[i+1] -=10; s[i]++; }
//output
for (int i=0;i<101;++i)
  cout << ( s[i]+ '0' );
cout <<endl;
```

۳. (متوسط) برنامه‌ای بنویسید که دو عدد صد رقمی از ورودی گرفته، حاصل ضرب آنها را خروجی دهد (ضرب دو عدد صد رقمی حداکثر ۱۹۹ رقمیست)

به مانند سوال قبل عمل می‌کنیم، منتها عملیات ضرب را انجام می‌دهیم (یک حلقه بیشتر لازم دارد و نرمالسازی آن تقسیم و باقیمانده به جای تفریق و جمع می‌طلبد) یعنی هر رقم از عدد دوم در تمامی رقم‌های عدد اول ضرب شده با حاصل جمع می‌شود :

```
char a[101], b[101] , s[200];
cin >>a>>b;
for (int i=0; i<200; ++i) s[i]=0;
for (int j=0; j<100; ++j)
  for (int i=0; i<100; ++i)
        s[ 100 + j+i]+= ( b[j] - '0' ) * ( a[i] - '0' ); //keep the first 100 for normalization!
//normalization:
for (int i=199;i>=0;--i)
  if (s[i]>=10) { s[i-1] += s[i] /10;  s[i]%=10; }
//output
for (int i=0;i<200;++i)
  cout <<( s[i] + '0' );
cout <<endl;
```

٤. (سخت) برنامه‌ای بنویسید که یک عدد صد رقمی و یک عدد ۱۰ رقمی از ورودی گرفته، حاصل باقیمانده اولی تقسیم بر دومی را خروجی دهد! (دقت کنید که همه اعداد ۱۰ رقمی در int جای نمی‌گیرند ولی در long long جای می‌گیرند، چه برسد به ۱۱ رقمی ها!)

به برنامه زیر دقت کنید:

```
long long a, t=0; //11 digits wont fit on int, so long long
char s[101];
cin >>s>>a;
for (int i=100;i>=0; --i)
{
  t = t * 10 + ( s[i] - '0' ) ;
  t = t % a;
}
```

از آنجایی که تنها باقیمانده را می‌خواهیم، لازم نیست اول کل عدد را حساب کنیم بعد باقیمانده بگیریم بلکه می‌توانیم کم کم رقم‌های عدد را بیافزاییم و باقیمانده بگیریم (رجوع شود به خواص عملگر باقیمانده در نظریه

اعداد)

۵. (خیلی سخت) برنامه‌ای بنویسید که یک عدد صد رقمی را بر یک عدد ۳۰ رقمی تقسیم کند و حاصل را خروجی دهد. (برنامه باید در کمتر از یک ثانیه جواب دهد)

شرط انتهایی برنامه یعنی با استفاده از تفریق ساده اینکار را نکنیم! چون اگر حاصل تقسیم مثلا ۱۰۰۰۰۰۰۰ باشد باید به این تعداد تفریق ساده انجام دهیم! اما نکته در آن است که حتی وقتی به صورت دستی دو عدد را تقسیم می‌کنیم، از تفریق استفاده می‌کنیم ولی نه تفریق ساده. به جای اینکه تفریق را از سمت راست انجام دهیم، از سمت چپ انجام می‌دهیم (لطفا بر روی کاغذ دو عدد را تقسیم کنید!) تفریق از سمت چپ در واقع معادل است با تفریق ۱۰ به توان ایکس تا از آن عدد (در مبنای ده آنقدر شیفت به چپ داده‌ایم تا زیر بزرگترین عدد قرار گیرد)

همان راه را در برنامه نیز باید انجام دهیم که حجم کد نسبتا زیادی خواهد داشت.

۶. (متوسط) رکوردی برای دانشجو داریم که دارای فیلدهای نام (۲۰ حرف) نام خانوادگی (۲۰ حرف) و شماره دانشجویی (عدد) است. اطلاعات صد دانشجو را در یک فایل باینری در قالب این رکورد داریم. می‌خواهیم این فایل را مرتب کنیم. درست است که کار با فایل در مقایسه کار با حافظه بسیار کند است، اما مشکل ما آنست که حافظه بسیار محدودی داریم و تنها دو متغیر r۱ , r۲ را از این رکورد در حافظه داریم (و دیگر نمی‌توانیم متغیری از این رکورد در حافظه بگیریم) بنابراین می‌خواهیم این فایل را مرتب‌سازی خارجی کنیم. مشخص کنید که کدام الگوریتم مرتب سازی برای اینکار بهتر است و چرا؟

مرتب‌سازی خارجی یعنی بدون اینکه محتوای فایل را داخل حافظه بیاوریم و مرتب کنیم و دوباره به فایل برگردانیم، فایل را مرتب کنیم. این کار لازم است چون اکثر فایلهای اطلاعات بسیار بزرگتر از کل حافظه هستند، و اگر نبودند چه احتیاجی بود اطلاعات در فایل ذخیره شود!

از آنجایی که عملیات داخل فایل بسیار کند است (حدود یک میلیون برابر کندتر از حافظه) باید الگوریتم مرتب سازی را انتخاب کنیم که کمترین میزان خواندن و نوشتن از فایل را انجام دهد. به عنوان مثال الگوریم حبابی تقریبا تمام جفت مقادیر را با هم مقایسه می‌کند، یعنی تقریبا هرکدام از مقادیر باید n بار از فایل خوانده شوند. اما الگوریتم انتخابی یا الگوریتم درجی یک عنصر را با تمام عناصر مقایسه کرده جای آنرا می‌یابد.

۷. (بسیار سخت) مرتب سازی سوال بالا را انجام دهید.

جهت انجام این مرتب سازی، کافیست سه تابع بنویسیم :

```
record get(int index)
```

که باید index امین رکورد داخل فایل را بخواند و در یک متغیر برگرداند.

```
void set( record r, index)
```

که باید رکورد index امی که ما به آن می‌دهیم درون فایل بنویسد، و

```
int compare (record r۱, record r۲)
```

که باید دو رکورد را با شرایطی که می‌خواهیم مقایسه کند و نتیجه را بازگرداند. پس از آن کافیست الگوریتم مرتب‌سازی مورد نظر، مثلا حبابی را بنویسیم:

```
int n=100;
record r1,r2;
for (int j=0; j<n −1; ++j)
  for (int i=0; i<n−j; ++i)
        if ( compare ( r1=get(i) , r2=get(i+1) ) )
              { set (r1, i+1); set(r2, i); }
```

و تمام!

۸. (بسیار سخت) برنامه‌ای بنویسید که n را از ورودی گرفته، سپس n+۱ عدد که ابعاد n ماتریس هستند ورودی بگیرد (به غیر از اولی و آخری هر عدد تعداد ستون‌های ماتریس i و تعداد سترهای ماتریس i+۱ است) اکنون می‌دانیم که می‌توانیم تمام این ماتریس‌ها را به ترتیب ضرب کنیم. ضرب ماتریس‌هایی با ابعاد MxL , LxQ تعداد MxLxQ عملیات لازم دارد و یک ماتریس با ابعاد MxQ می‌سازد. مشخص کنید که چگونه این ماتریس‌ها را پرانتز گذاری کنیم که تعداد کل ضرب‌های لازم برای ضرب همه ماتریس‌ها مینیمم شود! (بازگشتی)

یک مسئله بازگشتی معمول. کافیست مانند مسئله جایگشت‌ها، پرانتزها را در تمامی نقاط امتحان کنیم.

۹. (سخت) برنامه‌ای بنویسید که یک متن نسبتا طولانی از ورودی گرفته و آنرا Justify کند، یعنی طوری خروجی دهد که در هر سطر (غیر از سطر آخر) دقیقا ۸۰ کاراکتر وجود داشته باشد که کاراکتر آخر آن یا حرف آخر یک کلمه باشد و یا کاراکتر ـ باشد که برای دوقسمت کردن یک کلمه استفاده شده است.

به این دسته از مسائل کار با رشته‌ها، کد اسپاگتی می‌گویند. این دسته کدها بسیار ساده هستند ولی کمی طولانی و برنامه‌نویسی که تمیز و دقیق برنامه ننویسد، قطعا در نوشتن این گونه کد دچار مشکل خواهد شد و خطاهای بسیاری خواهد داشت. رعایت تمام اصول برنامه‌نویسی به خصوص استفاده از توابع برای اینگونه برنامه‌ها لازم است.

۱۰. (سخت) رکورد زیر را برای تعریف درخت معین می‌کنیم:

```
struct tree {
  int data;
  tree *right_child;
  tree *left_child;
};
```

برنامه‌ای بنویسید که به صورت بازگشتی عدد nام سری فیبوناچی را حساب کند. حال این برنامه را طوری تغییر دهید که همگام با پیمایش شدن درخت بازگشت توسط سیستم، درختی نیز توسط این برنامه ساخته شود و مقادیر مرتبط درون آن قرار گیرد. سپس با اتمام توابع درخت از حافظه پاک شود.

```
int fibo(int n , tree * node)
{
  if (n<2) return 1;
  //creating
  node->right_child = new tree;
  node->left_child = new tree;
  //assigning
  node->data = fibo (n-1 , node->left_child ) + fibo (n-2 , node->right_child );
  //deleting
  int r=node->data;
  delete node->right_child;
  delete node->left_child;
  return r;
}
```

۱۱.(متوسط) تابعی بنویسید که دو لیست پیوندی و یک عدد دریافت کند، لیست دوم را در موقعیت عدد در میان لیست اول درج (Insert) کند.

```
void insert (Cell * list1, Cell * list2 , int position)
{
  Cell * t=list1;
  //first traversing till we reach desired position
  for (int i=0;i<position;++i)
        t=t->next;
  //now backup where we must insert
  Cell * backup=t->next;
  t->next=list2; //inserted
  //now traversing list2 till its end
  t=list2;
  for (int i=0; t->next ; ++i)
        t=t->next;
  //attaching the backed up part here
  t->next=backup;
}
```

۱۲. (آسان) خروجی برنامه زیر چیست؟

```
unsigned int size = -۱;
int *p = new int [size];
printf ( "%ud\n",p);
```

سوال بسیار جالب و پر نکته و جذاب است :) متغیر size مقداری معادل بیشترین مقدار مثبتی که در ۳۲ بیت جای می‌گیرد دارد (زیرا بدون علامت است و اگر از صفر یکی کم کنیم می‌شود بزرگترین عدد) که معادل ٤GB است. از آنجایی که در سیستم ۳۲ بیت، کل حافظه به این اندازه است (و در واقع به خاطر همین اندازه پوینتر و اینت در آن ۳۲ بیت است) و از آنجایی که کل حافظه خالی نیست (!) خط دوم برنامه نمی‌تواند حافظه درخواست شده را تخصیص دهد. دستورات تخصیص حافظه در صورت عدم موفقیت صفر برمی‌گردانند (اشاره‌گر بن‌بست)

خط آخر مقدار آدرسی اشاره‌گر را خروجی می‌دهد!

۱۳.(متوسط) اشتباهات برنامه زیر چیست؟ اصلاح کنید و بفرمایید برنامه چه می‌کند؟

```
template <class T>
T& item( T*a, int index) { return *(a+index); }
int & plus_equals(int &a, int &b) { return a=b; }
int & xor_eqauls(int &a, int &b) { return a^=b; }
int sum()
{
  int size=۱۰۰۰;
  int a[size];
```

```
for (int i=۰ ; i<size ; ++i)
        item(a,i)=rand() ٪ i ;
for (int i=۰ ; i<size ; i+=۲)
        xor_equals ( xor_equals ( xor_equals ( item ( a , i ) , item ( a , i+۱ ) ) , item ( a , i ) ) , item ( a , i+۱ ) );
int sum =۰ ;
for (int i=۰ ; i<size ; i+=۲)
        plus_equals ( sum , item(a,i));
return sum;
}
```

قسمت‌های قرمز در برنامه نبودند که اشتباه بود!

از آنجایی که تابع item مانند کروشه برای دسترسی به عناصر آرایه کار می‌کند، باید مرجع برگرداند نه کپی (وگرنه اگر آنرا مساوی چیزی قرار دهیم، مانند جایی که تصادفی پر کرده‌ایم، آرایه اصلی پر نمی‌شود) دو تابع دیگر از نامشان پیداست که چه می‌کنند، یکی معادل عملیات =^ و دیگری =+ است! منتها تفاوت تابع با عملگر آنست که عملگر عملوندهای خود را در دو طرف می‌پذیرد ولی تابع در داخل پرانتز جلوی خود.

برنامه یک آرایه ۱۰۰۰ تایی از اعداد را تصادفی پر می‌کند، سپس جای اعداد زوج و فرد آنرا عوض می‌کند و نهایتا جمع اعداد در اندیس‌های زوج (فرد سابق) را خروجی می‌دهد.

۱۴.(آسان) برنامه‌ای بنویسید که حافظه پویای گرفته شده در برنامه زیر را آزاد کند!

```
int ****p=new int ***=new int ** = new int * = new int[۲];
```

از آنجایی که ٤ بار از دستور تخصیص حافظه استفاده کرده‌ایم، قاعدتا باید چهار بار نیز از دستور آزادسازی حافظه استفاده کنیم، اما با ترتیبی برعکس تخصیص (زیرا این حافظه‌ها تودرتو هستند و اگر اولی را آزاد کنیم دیگر مابقی گم می‌شوند)

```
delete [] ***p; //this is a pointer to an array of size 2
delete **p; //this is a pointer to an int *
delete *p; //this is a pointer to an int **
delete p; //this is a pointer to an int *** (i.e int *** *p)
```

- سوالات پایان ترم به این صورت نیستند بلکه معمولا معقول‌تر و کلاسیک‌تر هستند.
- این سوالات جهت تسلط به مفاهیم مطرح شده ارائه شده اند.

موفق و پیروز باشید