مبانی برنامه‌سازی (C)

نیم‌سال اول ۹۵-۹۴

دانشکده‌ی مهندسی کامپیوتر

آزمون پایان‌ترم | مدت امتحان: ۲ ساعت و نیم

با دقت بخوانید!

این کتابچه‌ی امتحانی شامل ۶ سؤال و ۱۲ صفحه است. سؤال اول شامل ۱۰ قسمت چهارگزینه‌ای بوده که در هر کدام از آن‌ها تنها یک جواب درست در بین گزینه‌ها وجود دارد. هر پاسخ درست به هر قسمت تستی ۳ امتیاز مثبت و هر پاسخ غلط به هر کدام از آن‌ها ۱ متیاز منفی در پی خواهد داشت. برای سؤالات تشریحی در نظر داشته باشید که جواب‌ها باید بر اساس هر آن‌چه که در صورت سؤال‌ها خواسته شده است، پاسخ داده شوند.

سؤال پنجم، سؤالی امتیازی است که می‌توانید در صورت وقت اضافی به آن پاسخ دهید. همچنین سؤال ششم مخصوص دانشجویان کامپیوتر می‌باشد. پاسخ‌گویی به این سؤال توسط دانشجویان سایر رشته‌ها نمره‌ی اضافی **نخواهد داشت.**

از صفحاتی که سفید هستند می‌توان به عنوان چرک‌نویس استفاده کرد. به همین ترتیب انتظار می‌رود در کادرهای مشخص‌شده برای جواب هر سؤال، تنها کد مربوط به آن سؤال، تمیز و با خطی خوانا (همراه به کامنت‌گذاری مناسب) نوشته شود. در غیر این‌صورت، مسئولیت عدم رعایت این موضوعات بر عهده‌ی خود شماست!

استفاده از کتاب و یادداشت‌های شخصی آزاد، امّا تبادل آن با دیگر دانشجویان در جلسه‌ی آزمون ممنوع است و در صورت مشاهده تقلّب محسوب می‌شود. همچنین مجاز به استفاده از هیچ‌گونه وسیله‌ی الکترونیکی در جلسه‌ی آزمون نمی‌باشید و در صورت مشاهده تقلّب گرفته خواهد شد.

اسم و شماره‌ی دانشجویی خود را در پایین این برگه بنویسید. از آن‌جایی که ممکن است برگه‌ها برای تصحیح از هم جدا شوند، در پایین برگه‌های دیگر نیز شماره‌ی دانشجویی خود را بنویسید. همچنین مدرّس گروه خود را در پایین این برگه مشخص بفرمایید. در آخر این‌که سعی کنید قبل از شروع یک نفس عمیق بکشید!

موفق باشید.

لطفاً در این قسمت چیزی ننویسید!

۱ (xx/۳۰)	۲ (xx/۱۵)	۳ (xx/۱۵)	۴ (xx/۲۰)	۵ (xx/۱۰)	۶ (xx/۲۵)

نام و نام‌خانوادگی: **شماره‌ی دانشجویی:**

مدرّس: ترکمن □ ریواده □ سپهری‌نور □ طاهرخانی □ فرجی‌پور □ غیبی □ مصطفی‌زاده □ هراتیان □

	۱	۲	۳	۴
۱.الف				
۱.ب				
۱.ج				
۱.د				
۱.ه				
۱.و				
۱.ز				
۱.ح				
۱.ط				
۱.ی				

سؤال ۱. سؤالات چهارگزینه‌ای (۳۰ نمره).

۱.الف. خروجی قطعه کد زیر چیست؟

```c
#include <stdio.h>

int main()
{
    char st1[]="Hello";
    char st2[10]={0};
    char *t, *s;
    s = st1;
    t = st2;
    while (*t=*s)
        *t++ = *s++;
    printf("%s\n", st2);
    return 0;
}
```

۱) ello ۲) Hello ۳) HelloHello ۴) خروجی ندارد.

۱.ب. خروجی قطعه کد زیر چیست؟

```c
#include <stdio.h>

int main()
{
    int x[5] = {12, 2, 11, 20, 25};
    int i, j, k;
    i = --x[1];
    j = x[1]++;
    k = x[++i];
    printf("%d, %d, %d", i, j, k);
    return 0;
}
```

۱) 3, 1, 20 ۲) 2, 2, 11 ۳) 2, 1, 11 ۴) 2, 1, 2

شماره دانشجویی:

۱.ج. اگر آدرس آرایه x از آدرس ۲۰۰۰ حافظه شروع شده باشد ، خروجی قطعه کد زیر چه خواهد بود؟

```
#include <stdio.h>

int main()
{
    int x[3][4] = {2, 6, 1, 3, 9, 5, 4, 7, 6, 2, 8, 5};
    printf("%u, %u, %u\n",x[0]+1, *(x[0]+1),*(*(x+0)+1));
    return 0;
}
```

۱) 2001, 6, 3 ۲) 2004, 2, 2 ۳) 2004, 6, 6 ۴) خطا رخ می‌دهد.

۱.د. خروجی قطعه کد زیر چیست؟

```
#include<stdio.h>
#define SIZE 10;

int main( void )
{
   int x[SIZE], j;
   for(j=0; j<SIZE; j++)
      x[j]=3+4*j;
   for(j=0; j<SIZE; j+=3)
      printf("%d, ", x[j]);
   return 0;
}
```

۱) 3, 15, 27, 39 ۲) 0, 21, 42, 63 ۳) 7, 19, 31, ۴) خطا رخ می‌دهد.

۱.ه. خروجی قطعه کد زیر چیست؟

```
#include<stdio.h>

int main()
{
    int a = 2, b = 3, result = 0;
    sum(a, b, result);
    printf("%d\n", result);
    return 0;
}
void sum(int a, int b, int result)
{
    result = a + b;
}
```

۱) 5 ۲) 0 ۳) عددی به‌جز 5 و 0 ۴) خطا رخ می‌دهد.

۱.و. با فرض اینکه پس از اجرای اولین خط از قطعه کد زیر آرایه a در آدرس ۱۰۰ حافظه ساخته شده باشد، پس از اجرای خط سوم، a به چه آدرسی اشاره می کند؟ (فرض کنید اعداد integer در ۲ بایت ذخیره می شوند)

```
int a[10], *p;
p = a;
a += 5;
```

۱) 105 ۲) 110 ۳) 108 ۴) هیچ‌کدام

شماره دانشجویی:

۱.ز. در سوال قبل، اگر به جای عبارت `a += 5`، عبارت `p += 5` را قرار دهیم، آنگاه پس از اجرای خط سوم p به چه آدرسی اشاره می کند؟

```
int a[10], *p;
p = a;
a += 5;
```

۱) 105 ۲) 110 ۳) 108 ۴) هیچ‌کدام

۱.ح. خروجی قطعه کد زیر چیست؟

```
#include<stdio.h>

int main()
{
    union var
    {
        int a, b;
    };
    union var v;
    v.a=10;
    v.b=20;
    printf("%d\n", v.a);
    return 0;
}
```

۱) 10 ۲) 20 ۳) 30 ۴) 0

۱.ط. خروجی قطعه کد زیر چیست؟

```
#include<stdio.h>

int main()
{
    enum days {MON=-1, TUE, WED };
    printf("%d, %d, %d \n", ++MON, TUE, WED);
    return 0;
}
```

۱) -1, 0, 1 ۲) 0, 1, 2 ۳) 0, 0, 1 ۴) خطا رخ می‌دهد.

۱.ی. خروجی قطعه کد زیر چیست؟

```
#include<stdio.h>
struct course
{
    int courseno;
    char coursename[25];
};
int main()
{
    struct course c[] = { {102, "Java"},
                          {103, "PHP"},
                          {104, "DotNet"}};
    printf("%d ", c[1].courseno);
    printf("%s\n", (*(c+2))->coursename);
    return 0;
}
```

۱) 104 DotNet ۲) 103 DotNet ۳) 103 PHP ۴) خطا رخ می‌دهد.

شماره دانشجویی:

سؤال ۲. تعداد کلمات (۱۵ نمره). برنامه‌ای بنویسید که یک رشته را در تابع اصلی از ورودی خوانده، سپس آن رشته را به تابعی دیگر ارسال کند. در آن تابع کلمات رشته به ترتیب از چپ به راست و هر کدام در یک خط مجزا چاپ شود و تعداد کلمات رشته محاسبه شده و در تابع اصلی چاپ شود. توجه داشته باشید که حداکثر طول رشته ورودی ۴۹ کاراکتر و کلمات رشته ورودی با یک (یا چند) کاراکتر '_' از یکدیگر جدا شده اند. به غیر از توابع scanf و printf مجاز به استفاده از هیچ تابع کتابخانه‌ای دیگری نیستید. به عنوان مثال برای ورودی "acd___dfg_sd_____nvm_" این خروجی چاپ خواهد شد:

```
acd
dfg
sd
nvm
4
```

یافتن تعداد کلمات:

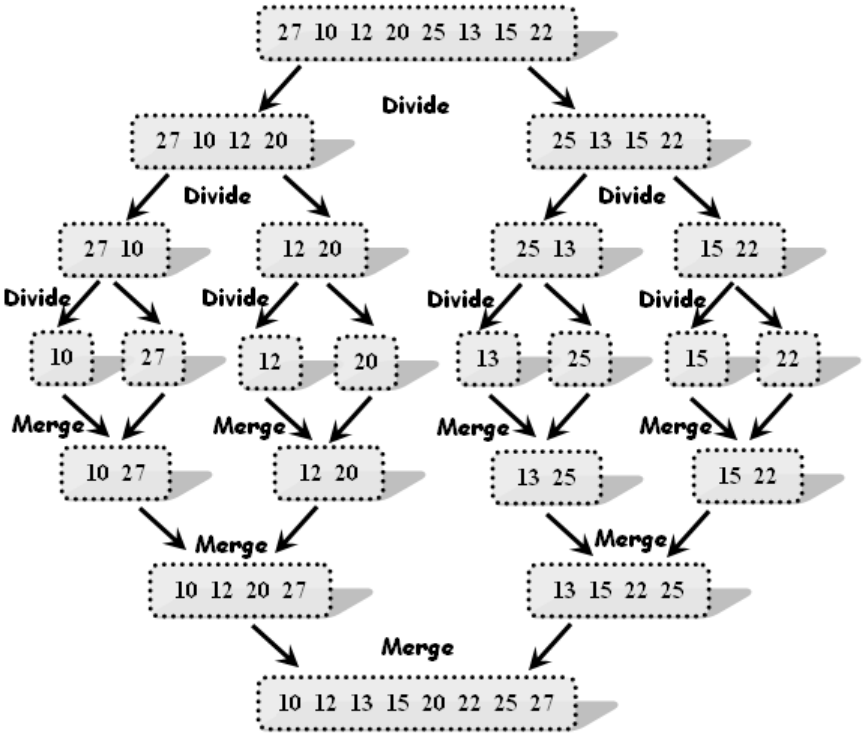
شماره دانشجویی:

سؤال ۳. یافتن مُد (۱۵ نمره). برنامه ای بنویسید که ۸۰ عدد بین ۰ تا ۹ را که به طور تصادفی روی یک خط تایپ شده اند از ورودی بخواند. سپس رقمی را که بیشتر از همه تکرار شده همراه با تعداد تکرار در خروجی چاپ کند (اگر چند رقم، بیشترین تعداد را داشتند، تمامی آن‌ها را چاپ کند).

یافتن مُد:

شماره دانشجویی:

سؤال ۴. مرتب‌سازی ادغامی (۲۰ نمره). کُپُل می‌خواهد دوستان خود را در صف به ترتیب قد قرار دهد. برای این منظور ابتدا آن‌ها را در صف قرار داده و به فکر می‌نشیند. ناگهان راه‌حلی ابتکاری به ذهنش می‌رسد. ابتدا دوستانش را به دو گروه با تعداد مساوی (البته اگر تعدادشان فرد بود یکی از گروه‌ها یک نفر بیشتر خواهد بود) تقسیم کرده و سپس هر گروه را تا زمانی که تعداد به دست آمده فقط یک نفر باشد تقسیم به دو دسته می‌کند. در روند عکس، گروه‌ها با هم به گونه‌ای که مرتب باشند، ادغام می‌شوند.



می‌دانیم که اعضای هر گروه به ترتیب قد مرتّب هستند. کُپُل در این فکر است که از این خاصیت استفاده کند و هر دو گروهی را تنها با خواندن یک‌باره‌ی آن‌ها (از اول تا به آخر) در یک گروهِ مرتب، ادغام کند. بنابراین برای ادغام این دو گروهِ مرتب، هیچ‌گاه از روش‌های مرتب‌سازی معمول (که هیچ فرضی بر روی داده‌ها ندارند) استفاده نمی‌کند.

تابعی بازگشتی بنویسید که این روند را پیاده سازی کرده و تمام دغدغه‌های کُپُل را مرتفع کند! سپس آن را در برنامه اصلی فراخوانی کنید. توجه داشته باشید که تعداد دوستان کُپُل را ندانسته و در زمان فراخوانی تابع آن را از کاربر دریافت می‌کنیم. از طرف دیگر اسم دوستان کپل فاقد اهمیت بوده و فقط ترتیب قد آن‌ها مهم است.

مرتب‌سازی ادغامی:

شماره دانشجویی:

ادامه‌ی مرتب‌سازی ادغامی:

شماره دانشجویی:

سؤال ۵. پرانتزگذاری صحیح (امتیازی) (۱۰ نمره) برنامه ای بنویسید که یک عبارت ریاضی دارای پرانتز را از ورودی دریافت کند و به تابع ارسال نماید. در این تابع باید پرانتز های عبارت ریاضی بررسی شود. صورتی که پرانتز ها درست قرار داده شده باشند عبارت«YES» و در صورتی که اشتباه قرار داده شده اند عبارت«NO» نمایش داده شود (توجه شود که در این سؤال تنها نحوه‌ی پرانتزگذاری صحیح مد نظر است و نه یک عبارت ریاضی صحیح. بنابراین چنین عبارتی $S(+۱)$ نیز یک پرانتزگذاری صحیح محسوب می‌شود. همچنین در هر عبارت ریاضی، تنها چهار عملگر اصلی، اعداد و حروف ظاهر می‌شوند). برای مثال:

input:	**output:**
((a+b)+c)	YES
((a+b)	NO
a+b))+((b+c	NO

پرانتزگذاری صحیح:

شماره دانشجویی:

ادامه‌ی پرانتزگذاری صحیح :

سؤال ۶. پیاده‌سازی بردار (۲۵ نمره). می‌خواهیم برای محاسبات برداری، کلاسی به نام Vector تعریف کنیم که قالب زیر را داراست:

```
#include<iostream>
using namespace std;

template<typename Type>
class Vector
{
    private:
        int dim;
        int capacity;
        Type* arr;
    public:
        Vector(){dim = 0; capacity = 1; this->arr = new Type[1];}
        Vector(int dim); // 5 points
        Type operator[](int i) const {return this->arr[i];}
        Type & operator [](int i) {return this->arr[i];}
        Type* begin() {return this->arr;}
        Type* end() {return this->arr + this->dim}
        int dimension() {return this->dim;}
        void insert(Type input, int index); // 5 points
        void remove(int index); // 5 points
        Vector<Type> operator+(const Vector<Type>& v2); // 10 points
};
```

هر بردار دارای سه عضو شخصی است. dim که نمایان‌گر بعد آن بردار (تعداد اعضا)، capacity نماینده‌ی ظرفیت آن بردار (حافظه‌ای که گرفته شده است) و arr هم آدرس شروع حافظه‌ی گرفته شده است. نکته‌ی قابل توجه این‌که capacity (یعنی حافظه‌ی گرفته شده) همواره برابر با کوچکترین توانی از دو است که از dim بزرگتر است. بنابراین در صورت اضافه شدن

شماره دانشجویی:

عضوی به بردار، اگر نیاز به افزایش ظرفیت بود، به میزان توان بعدی دو که در حال حاضر در capacity است، حافظه‌ای گرفته می‌شود.

با این مشخصات، برای کلاس بردار،

الف) سازنده‌ای که مقدار dim را ورودی می‌گیرد.

ب) تابع insert که مقدار input را در جایگاه index اضافه می‌کند (مراقب باشید که در صورت تکمیل ظرفیت، باید به ظرفیت بردار اضافه شود).

ج) تابع remove که مقدار input را از جایگاه index حذف می‌کند (مراقب باشید که در صورت کمتر شدن dim از capacity به اندازه‌ی توانی از دو، باید از ظرفیت بردار کم شود).

د) عملگر جمع را برای دو بردار سربارگذاری کنید. خروجی این جمع، حاصل جمع برداری دو بردار مورد نظر است. اگر دو بردار مذکور بعد یکسانی نداشتند، اعضای بردار با بعد کوچکتر که از بردار بزرگتر کم‌تر دارد را صفر در نظر می‌گیریم.

بردار:

شماره دانشجویی:

ادامه‌ی بردار:

خط پایان یا پایان خط!

شماره دانشجویی: