به نام خدا

دانشکده مهندسی برق و کامپیوتر — مبانی کامپیوتر و برنامه‌نویسی

پاسخ‌نامه‌ی تکلیف سوم پاییز ۹۲ — یخ تحویل: --

۱- خروجی تکه کدهای زیر را بنویسید.[طراح : مراحمی]

1) int x = 10,y; y = x++; cout<<y; 10	2) int x = 10,y; y = x++; cout<<x; 11	3) int x = 10; x++; cout<<x; 11
4) int x = 10,y; y = ++x; cout<<y; 11	5) int x = 10; cout<<++x; 11	6) int x = 10; cout<<x++; 10

** ۱) ابتدا x درون y ریخته شده و سپس یکی به x اضافه میشود

پس خروجی آن که y . است میشود ۱۱

۲) ابتدا x درون y ریخته شده و سپس یکی به x اضافه میشود

پس خروجی آن که x . است میشود ۱۱

۳) ابتدا یک عدد به x 11 چاپ ، ۱۱+۱= اضافه میشود،پس ۱۱

میشود.

٤) ابتدا یک عدد به x اضافه میشود سپس درون y ریخته میشود.

پس ۱۱ چاپ میشود.

٥) ابتدا یک عدد به x اضافه شده و چاپ میشود یعنی ۱۱ چاپ

میشود.

٦) ابتدا x چاپ میشود و سپس یکی به آن اضافه میشود،یعنی

همان ۱۱ چاپ میشود.

۲- برنامه‌ای بنویسید که یک عدد طبیعی از ورودی بگیرد و تعداد سال و ماه و روز معادل آن را نمایش دهد. در واقع عدد ورودی تعداد کل روزهای گذشته از اول یک سال مشخص را نشان می‌دهد و عدد خروجی نشان می‌دهد از ابتدای آن سال تاکنون چند سال و چند ماه و چند روز گذشته است. برای مثال اگر کاربر عدد ۷۹۶ را وارد کند، خروجی، ۲ سال و ۲ ماه و ٦ روز خواهد بود. برای سادگی همه سال‌ها را ۳٦۵ روزه در نظر بگیرید اما مراقب تفاوت تعداد روش‌های ماه‌های اول سال با دیگر ماه‌ها باشید. (استفاده از شرط مجاز نیست.)[طراح : مراحمی]

```
#include <iostream>
using namespace std;
int main()
{
	int num=0;
	cin>>num;
	int y = num/365;
	num = num - (y*365);
	int m= num/186;
	int tmp= m-1;
	tmp= m*30 - tmp*31;
	int m1 = m*6;
	m= ( (num % 186)/tmp );
	int d=num-((m*tmp) +(m1*31));
	m= m+m1;
	printf("year:%d month:%d day:%d",y,m,d);

}
```

۳- برنامه‌ای بنویسید که دو عدد را از ورودی بگیرد و مقادیر دو متغیر را با استفاده از متغیر سوم جابه‌جا کند(این عمل جابه‌جایی را در اصطلاح Swap گویند.). کد لازم برای جابه‌جایی دو متغیر val1 و val2 را به جای توضیحات (رنگ سبز) در کد زیر قرار دهید. برای جابه‌جایی از متغیر temp کمک بگیرید. [طراح: مراحمی]

```
int main(){
int val1=14;
int val2=22;
int temp;
     printf(“before swapping: val1 = %d ,val2 = %d\n”,val1,val2);

     temp = val1;
     val1 = val2;
     val2 = temp;

     printf(“after swapping: val1 = %d ,val2 = %d\n”, val1,val2);
return 0;
}
```

٤- با توجه به کد زیر دلیل استفاده از ثابت‌ها را توضیح دهید.[طراح : خوبی]

```c
#include<stdio.h>
int main()
{
        const int RATE=100;
        int hour,day,sum;
        sum=0;
        printf("Hi first Worker! Enter hour of your work in day:");
        scanf("%d",&hour);
        printf("Hi! Enter number of day you are working in month:");
        scanf("%d",&day);
        sum = hour * day * RATE;
        printf("\nyour payroll is %d$\n",sum);

        sum=0;
        printf("Hi secend Worker! Enter hour of your work in day:");
        scanf("%d",&hour);
        printf("Hi! Enter number of day you are working in month:");
        scanf("%d",&RATE);
        sum = hour * day * RATE;
        printf("\nyour payroll is %d$\n",sum);
        return 0;
}
```

استفاده از متغییر به جای عدد ثابت به ما این امکان را می‌دهد تا برنامه‌هایی با قابلیت استفاده مجدد بنویسیم و بتوانیم بعدها به سادگی در آنها تغییر ایجاد کنیم.

اما مشکلی که استفاده از متغییرها به دنبال دارد امکان تغییر ناخواسته در بین برنامه است که استفاده از متغییرهای ثابت این مشکل را برطرف می‌کند.

در پاسخ کافیست نوشته شود ثابت تعریف کردن Rate موجب می‌شود اگر در برنامه به طور ناخواسته قصد تغییر آن را داشته باشیم با خطای زمان کامپایل مواجه شویم.

**زیرا در خط ۱۸ از کارگر خواسته شده تا تعداد روزهایی که به کار مشغول بوده را وارد کند تا برنامه آن را در متغیر day توسط تابع scanf قرار دهد،اما برنامه نویس در اثر اشتباه و فراموشی در تابع scanf به جای قرار دادن متغیر day متغیر rate را قرار داده است که در ابتدای کد آن را از نوع ثابت و به منظور نرخ روزانه کار قرار داده ایم که جمع طبق فرمول داده شده محاسبه شود.پس اگر از نوع ثابت آن را معرفی نکرده بودیم مقدار آن از ۱۰۰ به تعداد روزهای کارگر تغییر میکرد و حقوق اشتباه محاسبه میشد.

ضمن اینکه مقدار day نیز برابر مقدار day کارگر اول میشد.

ولی با اینکار کامپایلر ارور داده و برنامه نویس متوجه اشتباه خود میشود.

۵- در هر مورد تفاوت خروجی برنامه داده شده را بررسی کنید و دلیل بیاورید که چرا هر یک از خروجی‌ها مطلوب و یا نامطلوب است:[طراح : خوبی]

```c
1- int a=10,b=4;
   printf("a/b = %d.\n",a/b);
```

```
printf("a/b = %f.\n",a/b);
```

خروجی قسمت اول:۲

خروجی قسمت دوم: بسته به کامپایلر ۰ یا اعداد بی ربط

**با تقسیم ۱۰ بر ٤ انتظار داریم ۲.۵ چاپ شود در صورتی که اولی ۲ و دومی ۰ چاپ میکند. در اولی با ٪d خروجی را به صورت صحیح خواستیم.

پس فقط قسمت صحیح آن چاپ میشود. خروجی printf دوم را از نوع float تعیین کرده ایم پس به حاصل تقسیم به صورت float نگاه میکند که عدد بسیار کوچکی است و آن را چاپ میکند.

```
2- int a=-45;
unsigned b=a;
   printf("a is %d.\n",a);
   printf("b is %d.\n",b);
```

خروجی قسمت اول: -٤٥

خروجی قسمت دوم: بسته به کامپایلر -٤٥ یا اعداد بی ربط یا خطا

**هردو خروجی مطلوب هستند،زیرا متغیر a برابر ٤٥ – قرار داده شده و در تابع printf اولی متغیر a را به صورت ٪d یعنی همان عدد صحیح فراخوانی کرده ایم،پس همان ٤٥ – نمایش داده میشود که مطلوب است.در تابع printf دومی متغیر b را به صورت ٪d که عدد صحیح است فراخوانی کرده ایم و از آنجا که تعداد بیت های یکسانی به unsigned و int اختصاص داده میشود و نحوه ی بازخوانی بیت های آن ها مهم است و چون به صورت ٪d فراخوانی کرده ایم پس همان ٤٥ – چاپ میشود که مطلوب است.

```
3- int a=5,b=0;
float c=5,d=0;
printf("result of float var is %f.\n",c/d);
   printf("result of int var is %d.\n",a/b);
```

خروجی قسمت اول: بی نهایت

خروجی قسمت دوم: خطای زمان اجرا – بیرون آمدن از برنامه

**در اصل تقسیم یک عدد بر صفر تعریف نشده است و در اعمال ریاضی واعداد صحیح اشتباه است ولی از آنجا که در float، exception هایی تعریف شده است پس مقدار بی نهایت قابل چاپ است.

پس در تابع printf اول که نتیجه را به صورت ٪f فراخوانی کرده ایم یعنی float و مقدار مورد انتظار ما در این مورد بی نهایت است به صورت بی نهایت نیز(inf) چاپ میشود پس مطلوب است.

و در تابع printf دوم که نتیجه را به صورت ٪d یعنی عدد صحیح فراخوانی کرده ایم و از آنجا که در سیستم اعداد صحیح int بی نهایت نشان داده نمیشود و تقسیم بر ۰ ارور دارد پس سیستم در واقع ارور داده (integer by zero) پس مطلوب

است.

٦- اگر متغیری را در برنامه تعریف کنیم، اما آن را مقدار دهی نکنیم انتظار دارید چه مقداری در آن باشد؟ (این سوال تحقیق نیست و می‌خواهیم تصور و فکر خودتان را بیان کنید)

هر جوابی که دارای **استدلال منطقی** باشد مورد قبول است.

**یک حالت اینکه اگر چیزی در حافظه اختصاص یافته به آن متغیر نباشد کامپایلر مقدار آن را در صورت فراخوانی صفر در نظر بگیرد.

-۲ ممکن است از قبل مقادیری در حافظه ی اختصاص یافته به آن متغیر وجود داشته باشد،پس آن مقدار طبق فراخوانی ما تبدیل به عدد شده و نمایش داده میشود و یا ممکن است ما متغیر را int معرفی کرده باشیم و بعد از آن متغیر دیگری از نوع float معرفی کنیم،لذا در صورت فراخوانی مجدد متغیر int که مقداردهی نکرده ایم با توجه به اینکه خانه های حافظه هر کدام آدرس مخصوصی دارند چند بیت) ۸ بیت(اول متغیری که به صورت float معرفی شده را خوانده و چاپ کند و یا هر متغیری که بعد از آن معرفی کرده ایم.

-۳ حالت دیگر این است یک عدد به صورت رندم چاپ شود،البته این هم بستگی به مقادیری دارد که از قبل در حافظه ی اختصاص یافته به متغیر وجود داشته است. از جواب های کامل: ۱) متغیر num3 به صورت unsigned معرفی شده است ولی نوع آن در تابع scanf به صورت ٪d که int است معرفی شده و دریافت می شود.

۲)متغیر num2 که از نوع unsigned است برابر متغیر num1 که از نوع int است قرار داده شده است یعنی متغیر num1 درون num2 ریخته شود که در این صورت اگر num1 یک عدد منفی انتساب داده شده باشد نمی تواند درون یک متغیر unsigned قرار داده شود،البته مهم نحوه ی باز خوانی آن است،یعنی به یک شکل ذخیره میشوند.

۳)متغیر num3 درون متغیر num1 ریخته شده است که یکی int و unsigned است.

تذکر:البته با توجه به اینکه متغیر num1 و num3 به صورت ٪d در تابع scanf دریافت شده اند و نیز هر مقدار unsigned می تواند در int قرار بگیرد مشکلی پیش نمی آید.

٤) در خط ۷ یک متغیر int و یک unsigned با هم جمع شده اند.

تذکر:البته برنامه بدون ارور کامپایل شده و کار میکند.زیرا در تابع scanf دو متغیر به صورت ٪d دریافت شده است.ونیز تفاوت تعریف unsigned و int بیشتر در مقایسه دو عدد است و خوانده شدن عدد و محاسبه بستگی به دید ما نسبت به بیت های ذخیره شده دارد که چه نوعی آن ها را بخوانیم.

۷- تمام اشکالات احتمالی کد زیر را به تفکیک بیان کنید.[طراح: بهرامی]

```
intnum1,sum;
unsigned num2,num3;
printf(“Enter 2 numbers:\n”);
scanf(“%d%d”,&num1,&num3);
num2=num1;
num1=num3;
sum=num1+num2;
```

```
printf(“sum of your numbers is %d.\n”,sum)
```

** ۱) متغیر num3 به صورت unsigned معرفی شده است ولی نوع آن در تابع scanf به صورت ٪d که int است معرفی شده و دریافت می شود.

۲)متغیر num2 که از نوع unsigned است برابر متغیر num1 که از نوع int است قرار داده شده است یعنی متغیر num1 درون num2 ریخته شود که در این صورت اگر num1 یک عدد منفی انتساب داده شده باشد نمی تواند درون یک متغیر unsigned قرار داده شود،البته مهم نحوه ی باز خوانی آن است،یعنی به یک شکل ذخیره میشوند.

۳)متغیر num3 درون متغیر num1 ریخته شده است که یکی int و unsigned است.

تذکر:البته با توجه به اینکه متغیر num1 و num3 به صورت ٪d در تابع scanf دریافت شده اند و نیز هر مقدار unsigned می تواند در int قرار بگیرد مشکلی پیش نمی آید.

٤) در خط ۷ یک متغیر int و یک unsigned با هم جمع شده اند.

تذکر:البته برنامه بدون ارور کامپایل شده و کار میکند.زیرا در تابع scanf دو متغیر به صورت ٪d دریافت شده است.ونیز تفاوت تعریف unsigned و int بیشتر در مقایسه دو عدد است و خوانده شدن

عدد و محاسبه بستگی به دید ما نسبت به بیت های ذخیره شده دارد که چه نوعی آن ها را بخوانیم.

تحقیق:

۸- انواع خطا را با توجه به آنچه در کلاس آموخته‌اید توضیح دهیدو برای هر کدام مثالی بیابید. (تکه کدهای مشابه در تکلیف افراد نمره نخواهند گرفت پس خودتان فکر کرده و از جستجو نیز کمک بگیرید)[طراح : خوبی]

انواع خطا عبارت است از : زمان کامپایل (نحوی – نوع داده) و زمان اجرا

مثال ها:

نحوی : همان خطای دستوری است مثلا main را بنویسیم mein.

نوع داده: در تعداد و یا نوع آرگومان های تابع مغایرتی باشد مثلا تابع دو متغیره را با سه متغیر فراخوانی کنیم.

زمان اجرا: تقسیم بر صفر – کمبود حافظه

**

Syntax Error:

```
cout >> a;
```

وقتی که شیوه نوشتن دستور طبق استاندارد های تعریف شده، نباشد.

Type Checking error:

```
int find(double n)
{
    ❊/some code/❊
}
int main()
{
    int a;
    cin >> a;
```

```
        find(a);
}
```

وقتی که تابع انتظار یک نوع متغیر داشته باشد اما متغیری از نوع دیگر برای آن ارسال شود.

Runtime Error:

```
int n;
int a[500];
cin >> n;
a[n]=2000;
```

(در این مثال ممکن است n گرفته شده بیشتر از ۵۰۰ باشد)

هنگامی که در زمان اجرای برنامه اشکالی غیر منتظره، مانند بیرون رفتن از آرایه، پر شدن حافظه، تقسیم بر صفر یا غیره اتفاق بیافتد.

۹- درباره تفاوت دو کاراکتر کنترلی CR(Carriage Return)و LF(Line Feed)تحقیق کنید و مثال بیاورید. (برای مثال سیستم عامل‌های مختلف رفتار متفاوتی نسبت به این کاراکترها دارند)[طراح : مراحمی]

کاراکتر CR و LF با کدهای 10 و 13 دسیمال ('\r' , '\n') ، به ترتیب باعث حرکت مکان نما به شروع خط جاری و خط بعد می شود. این دو کاراکتر کنترلی برای رفتن به خط بعد مورد استفاده قرار میگیرند. سیستم عامل های مختلف راه های متفاوتی در این زمینه دارند. مثلا Mac '\r' را به عنوان خط جدید و Unix و Linux '\n' را میشناسند. ویندوز هر دوی اینها را میشناسد. به همین دلیل است که فایل نوشته شده با یک سیستم عامل ممکن است با سیستم عامل دیگر به درستی باز نشود.

۱۰- آیا می‌توان نماد انتگرال (∫) را با استفاده از ۲ کد اسکی نمایش داد؟ برنامه‌ای بنویسید که این عبارت را نشان دهد. [طراح : بهرامی]

```
#include <stdio.h>

int main()
{
        printf("%c\n%c\n",244,245);
        return 0;
}
```

۱۱- در مورد Unicode و Encodingهای مختلف آن مثل UTF-8، UCS-2، UTF-16 و ... تحقیق کنید. هر یک برای ذخیره‌سازی یک کاراکتر از چند بیت استفاده می‌کنند. تفاوت آن‌ها با اسکی چیست؟ [طراح: آقاسی]

**یونیکد مانند اسکی استانداردی برای کد کردن کاراکتر هاست. در ابتدا هماهنگ با مجموعه جهانی کاراکتر ها توسعه یافت و استاندارد آن در کتابی با نام The Unicode Standard منتشر شد. آخرین استاندارد آن نسخه از آن شامل ۱۱۰۰۰۰ کاراکتر است که شامل کد برای مراجع تصویری، کدگذاری کاراکتر ها، مجموعه از فایل های کامپیوتری مرجع و شماری دیگر از اطلاعات مرتبط مانند اطلاعات کاراکتر ها، رندر کردن، نرمال کردن، مقایسه و ترتیب نمایش زبان های راست به چپ مانند عبری، فارسی و عربی است.

یونیکد می تواند با استفاده از روش های متفاوتی ساخته شود که مشهور ترین آنها UTF-8 و UTF-16 است. البته از USC-2 نیز که نسخه قدیمی تر است می توان استفاده کرد.UTF-8 برای هر کد اسکی، از یک بایت استفاده میکند که در هر دو شیوه کد نگاری کد یکسانی دارند.USC-2 از یک کد ۱۶ بیتی استفاده میکند اما نمیتواند تمامی کاراکتر هایی که در آخرین استاندارد یونیکد پشتیبانی میشود را رمزنگاری کند.UTF-16 نسخه ارتقاء یافته USC-2 است که از یک کد ۲۳ بیتی، استفاده میکند.

از تفاوت آن با اسکی میتوان به پشتیبانی از زبان های بالا به پایین و راست به چپ اشاره کرد.

Source: en.wikipedia.org/wiki/Unicode