ایمان محمدی شماره دانشجویی : ۹۹۱۰۲۲۰۷

تحلیل قطعه کد اول :

ابتدا i را یک متغیر از جنس عدد صحیح تعریف کرده و مقدار آن را برابر ۵ قرار میدهیم.

سپس در خط بعد، پوینتری از نوع void تعریف میکنیم.

سپس در خط بعد، آدرس مکان i در حافظه را در آن میریزیم و برابر با آن قرار میدهیم.

حالا در خط آخر دستور printf داریم. نکته این سوال این است که متغیر پوینتری از جنس void میتوانند به هر نوع متغیری اشاره کنند ولی باید قبل از مقداردهی، نوع آن ها را به نوع متغیری که به آن اشاره میکند، cast کرد که در اینجا این اتفاق نیفتاده و در نتیجه ارور کامپایل میدهد.

نحوه درست نوشتن کد در اینجا : * (int *) vptr

تحلیل قطعه کد دوم :

در خط اول ابتدا یک پوینتر به متغیری از جنس int به نام container تعریف میکنیم.

در خط دوم با استفاده از تابع malloc، ۲۰ بایت از فضای حافظه را اختصاص میدهیم و یک پوینتر از جنس void به اول آن برگردانده میشود. حالا این پونتر را به نوع (* int)، کست میکنیم و آن را داخل پوینتر container میریزیم و مقداردهی میکنیم پوینتر container را. در نتیجه آدرس اولین خانه حافظه را داخل پوینتر container ریخته ایم.

در خط بعد با استفاده از دستور printf، سایز پوینتر container را چاپ میکنیم که این عدد همواره ثابت نیست. در سیستم عامل ۶۴ بیتی این عدد، ۸ بایت است و در سیستم عامل ۳۲ بیتی، برابر با ۴ بایت میباشد. در نتیجه در سیستم عامل ۶۴ بیتی، عدد ۸، و در سیستم عامل ۳۲ بیتی، عدد ۴ چاپ میشود.

در خط آخر، با استفاده از تابع free، فضای اختصاص داده شده را آزاد میکنیم.

تحلیل قطعه کد سوم :

در خط اول ابتدا یک پوینتر به کاراکتر به نام a تعریف میکنیم که به "Respuesta incorrecta" اشاره میکند.

در خط بعد دستور printf داریم و وارد آن میشویم. در آخر عبارت اول ۱+ داریم در نتیجه عبارت اول از حرف دوم به بعد چاپ میشود و حرف اول که '%' است، چاپ نمیشود. سپس عبارت چاپ میشود تا زمانی که به %s میرسیم. با توجه به مقداری که به %s نسبت داده شده است، یعنی همان a، عبارت "Respuesta incorrecta" چاپ میشود به جای %s. بعد از آن دیگر هیچ %s ای در عبارت نداریم در نتیجه عبارت "Lorem Ipsum \0\b dolor sit amet" اصلا چاپ نمیشود.

در نتیجه با اجرای برنامه، خروجی ما هستش :

```
creías haber descubierto el plan? Respuesta incorrecta la posición se perdió!
```

تحلیل قطعه کد چهارم :

در خط اول یک پوینتر به کاراکتر به نام str تعریف میکنیم و به آن خانه اول استرینگ "Que" را نسبت میدهیم.

در خط بعد متغیر i را از جنس عدد صحیح تعریف میکنیم. حالا وارد حلقه میشویم. مقدار i در ابتدا برابر با ۰ است. حالا با توجه به دستور strlen(str) چک میکنیم که آیا مقدار i از طول str کمتر است یا نه. در ابتدا طول str، ۳ است. در نتیجه وارد حلقه میشویم. در مرحله اول ابتدا "Que" چاپ میشود. حالا پوینتر str یکی زیاد میشود. در نتیجه str به "ue" اشاره میکند. دوباره شرط حلقه را چک میکنیم برای مرحله بعدی. این بار i برابر است با ۱ با توجه به دستورات داخل حلقه و اینکه strlen(str) برابر با ۲ است؛ دوباره وارد حلقه میشویم، این بار "ue" چاپ میشود و حالا پوینتر str یکی زیاد میشود در نتیجه str به "e" اشاره میکند. دوباره شرط حلقه را چک میکنیم برای مرحله بعدی. این بار i برابر است با ۲ با توجه به دستورات داخل حلقه و اینکه strlen(str) برابر با ۱ است؛ وارد حلقه نمیشویم و برنامه با دستور return 0، به پایان میرسد.

در نتیجه با اجرای برنامه، خروجی ما هستش : Queue

تحلیل قطعه کد پنجم :

در خط اول یک پوینتر به کاراکتر به نام string تعریف میکنیم.
در خط بعد مقدار این پوینتر را برابر اولین خانه ای که "Le Casa de Papel" در آن ذخیره شده است قرار میدهیم. در تعریف استرینگ، اینجا استرینگ به صورت const ذخیره میشود و در نتیجه نمیتوان بعدا مقدار دیگری را در آن قرار داد. در نتیجه وقتی در خط بعد تغییر حرف ممکن نیست و در نتیجه به ارور کامپایل بر میخوریم.

تحلیل قطعه کد ششم :

در خط اول آرایه ای از پوینتر ها ایجاد میکنیم. داخل آن آرایه سه پوینتر به سه استرینگ "Helsinki", "Berlin", "Rio" قرار میدهیم.
در سه خط بعدی، سه اشاره گر i و j و k تعریف میکنیم که به سه خانه آرایه مان اشاره میکنند.
i و k و j اشاره گرهایی به خانه های آرایه هستند که خود اشاره گری به اول کلمات هستند در نتیجه i* و j* و k* اشاره گر هایی به اول کلمات مشخص شده هستند. در نتیجه i و j** و k** برابر حروف اول کلمات هستند.**
حالا در خط آخر وقتی میخواهیم i و j** و k** را چاپ کنیم، "HBR" چاپ میشود و سپس به خط بعدی میرود با دستور "\n". و در آخر برنامه با دستور return 0، به پایان میرسد.**

در نتیجه با اجرای برنامه، خروجی ما هستش : HBR

تحلیل قطعه کد هفتم :

در خط اول یک پوینتر به int به نام pointer تعریف میکنیم و نیز یک عدد صحیح به نام i تعریف میکنیم و مقدار ۲۲۲۶۵۴۸۴۲۱۳ را داخل آن میریزیم ولی نکته این است که به خاطر اینکه i یک int است و توانایی ذخیره عددی به آن بزرگی را ندارد، overflow رخ میدهد و یک عدد دیگر (۷۹۰۶۴۷۷۳۳) در i ذخیره میشود. در خط بعد آدرس متغیر i را در pointer میریزیم.

حالا در خط بعد متغیر دیگری به نام void_ptr از نوع اشاره گر جنس void تعریف میکنیم و آدرس متغیر pointer را در آن میریزیم.

حالا در خط بعد با دستور printf، باید

(\nValue of iptr = %d ", **(int **********)void_ptr)

را چاپ کنیم که void_ptr(******** int)** به %d نسبت داده شده است.**

در داخل این دستور، ابتدا پوینتر void_ptr را بهint ********** ، cast کرده ایم و مقداری که در void_ptr* است را می خواهیم که از آنجا که در خط های قبلی داشتیم void_ptr = &pointer در نتیجه void_ptr* در واقع همان pointer است و void_ptr**همان مقداریست که در i داریم و از آنجا که به int ********** ، cast کرده ایم و سایز این نوع از متغیر نیز مانند int چهار بایت است پس تابع printf می تواند داده درون آن را (که i هست) به عنوان int بخواند و چاپ کند.**

پس خروجی نهایی ما برای %d برابر با مقدار i است و داشتیم مقدار i برابر است با ۷۹۰۶۴۷۷۳۳

در نتیجه با اجرای برنامه، خروجی ما هستش : Value of iptr = 790647733