| | | |
|---|---|---|
| دانشگاه صنعتی خواجه نصیرالدین طوسی<br>K. N. TOOSI UNIVERSITY OF TECHNOLOGY | **دکتر نیک انجام**<br>**دکتر نصیحت کن** | **مبانی برنامه سازی - پاییز ۹۶**<br>**امتحان پایانترم** |
| **نام:** | **شماره دانشجویی:** | **بهمن ۹۶** |

- در سوالات برنامه نویسی ستونهای عمودی قرار داده شما که برای رعایت کردن تورفتگی ها (indentation) در برنامه های شماست. **در صورت رعایت نکردن تو رفتگی ها قسمتی از نمره کسر خواهد شد.**

**سوال ۱-** می خواهیم تابعی به نام capitalize بنویسیم که دو رشته حرفی را به صورت آرایه ای از کاراکترها به عنوان آرگومان بگیرد و اولی را در دومی کپی کند به صورتی هر کلمه در رشته دوم با حروف بزرگ شروع شود. تعریف کلمه دنباله ای از حروف انگلیسی است که پشت سر هم هستند و با کاراکترهای غیر حرفی از هم جدا می شوند. در زیر مثالی از استفاده از تابع capitalize را به همراه خروجی برنامه می بینید. تمام کاراکترهای رشته دوم همان کاراکترهای رشته اول است. فقط کاراکتر های شروع هر کلمه در رشته دوم با حروف بزرگ نوشته شده.

```
int main() {
  char s1[] = "salaam, be emtehan+!khoSH Amadid!!!";
  char s2[100];

  capitalize(s2,s1);

  printf("%s\n", s1);
  printf("%s\n", s2);

  return 0;
}
```

**خروجی:**

```
salaam, be emtehan+!khoSH Amadid!!!
Salaam, Be Emtehan+!KhoSH Amadid!!!
```

- بدنه تابع capitalize را بنویسید. **(۱۰ نمره، زمان ۱۵ دقیقه)**

```
void capitalize(char *dest, char *src) {
  int isletter = 0;

  while (*src != '\0') {
    char c = *src++;

    if (!isletter && c >= 'a' && c <= 'z')
      c += 'A' - 'a';

    *dest++ = c;

    isletter = (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z');
  }

  *dest = '\0';

}
```

**سوال ۲-** فرض کنید قطعه کد زیر در تابع main اجرا شده است:

```
struct PayTime {
  int hour;
  int min;
  struct PayTime *prev;
};

int main() {

  struct PayTime times[8];
  struct PayTime *p, *q;
  int i;

  times[0].min =  0;
  times[0].hour = 9;
  times[0].prev = NULL;
  p = &times[0];

  for (i = 1; i < 8; i++) {
    times[i].min = times[i-1].min + 40;
    times[i].hour = times[i-1].hour;
    times[i].prev = p;

    if (times[i].min >= 60) {
      times[i].min -= 60;
      times[i].hour += 1;
      p = &times[i];
    }

  }
```

اگر هر کدام از دستورات سمت چپ جدول زیر را در ادامه دستورات قبل اجرا کنیم چه چیزی چاپ می شود. اگر دستورات خطای کامپایل دارد جلوی آن علت را ذکر نمایید. **(۱۵ نمره، زمان ۲۰ دقیقه)**

- توجه کنید که برای یک اشاره گر به ساختار دو دستور زیر معادلند:

p->min
(*p).min

| | |
|---|---|
| `printf("%02d:%02d\n", times[0].hour, times[0].min);` | `09:00` |
| `for (i = 0; i < 8; i++)`<br>`  printf("%02d:%02d\n",times[i].hour,times[i].min);` | `09:00`<br>`09:40`<br>`10:20`<br>`11:00`<br>`11:40`<br>`12:20`<br>`13:00`<br>`13:40` |
| `q = &times[4];`<br>`q--;`<br>`printf("%02d:%02d\n", q->hour, q->min);` | `11:00` |
| `q = &times[4];`<br>`printf("%02d:%02d\n", q[2].hour, q[2].min);` | `13:00` |

| | |
|---|---|
| `q = times;`<br>`q += 4;`<br>`printf("%d\n", q-&times[1]);` | `3` |
| `for (i = 1; i < 8; i++)`<br>`  printf("%02d:%02d\n",times[i].prev->hour,`<br>`  times[i].prev->min);` | `09:00`<br>`09:00`<br>`10:20`<br>`11:00`<br>`11:00`<br>`12:20`<br>`13:00` |
| `printf("%02d:%02d\n",times[3]->hour,times[3]->min);` | `Compile error`<br>`times[3] is not a pointer, the operator -> cannot be used` |
| `printf("%02d:%02d\n", p.hour, p.min);` | `Compile error`<br>`p is pointer to struct`<br>`the operator "." cannot be used` |
| `printf("%02d:%02d\n", p->hour, p->min);` | `13:00` |
| `for (q = p; q != NULL; q = q->prev) {`<br><br>`  printf("%02d:%02d\n", q->hour, q->min);`<br><br>`}` | `13:00`<br>`12:20`<br>`11:00`<br>`10:20`<br>`09:00` |
| `q = p->prev->prev->prev;`<br>`printf("%02d:%02d\n", q->hour, q->min);` | `10:20` |

**سوال ۳-** فرض کنید به شما یک فایل به نام studentList.txt داده شده که در آن لیست تمام دانشجویان دانشگاه خواجه نصیر نوشته شده است. این فایل یک فایلی متنی است و در هر خط از فایل ابتدا نام کوچک، سپس نام خانوادگی و سپس شماره دانشجویی دانشجویان نوشته شده. محتویات چند خط اول فایل به صورت زیر است.

```
Akram    Zamani    9612345
Kamran    Borooji    9634253
Ahmad    Saeedi     9612444
Ramin    Saeedi     9694835
Pavin    Etesami     9619877
Pavin    Etesami     9619876
Abbas    Daee      9619871
Korosh    Daee       9685734
Paviz    Etesami    9618876
```

این فایل دو مشکل دارد. اول اینکه دانشجویان به ترتیب خاصی در این لیست چیده نشده اند. دوم اینکه فایل به صورت متنی است، در صورتی که ما می خواهیم فایل به صورت دسترسی اتفاقی (random access) باشد و طول هر رکورد ثابت باشد.

برای رفع این مشکلات ابتدا ساختاری به نام **Student** به صورت زیر تعریف می کنیم:

```
struct Student {
  char fname[20];
  char lname[20];
  int  id;
};
```

که در آن fname و lname و id به ترتیب نام کوچک، نام خانوادگی و شماره دانشجویی یک دانشجو را نشان می دهند.
سپس برنامه زیر را می نویسیم:

```
int main() {
  struct Student students[40000];
  int n;

  n = readStudents(students);

  sortStudents(students, n);

  writeStudents(students,n);

  return 0;
}
```

در این برنامه ابتدا یک آرایه با **حداکثر طول** 40000 تعریف شده که هر عضو آن از نوع struct Student است. سپس تابع readStudents اطلاعات مربوط به دانشجویان را خط به خط از فایل studentList.txt می خواند و در آرایه students می ریزد. همچنین به عنوان مقدار برگشتی تعداد دانشجویان در فایل studentList.txt را بر می گرداند. دقت کنید که عدد 40000 حداکثر تعداد دانشجویان است و نه تعداد واقعی دانشجویان. تابع sortStudents آرایه students را به ترتیبی که گفته خواهد شد مرتب می کند و در نهایت تابع writeStudents این آرایه را به صورت دسترسی تصادفی (بایت به بایت) توسط تابع fwrite در فایلی با نامه studentList.bin ذخیره می کند. این سوال ۴ قسمت دارد که به هر کدام از آنها می توانید مستقل از جواب دادن به قسمت های دیگر، جواب دهید.

**الف)** تابع readStudents را بنویسید **(۱۰ نمره، زمان ۱۰ دقیقه)**:

```
int readStudents(struct Student students[]) {
  FILE *f = fopen("studentlist.txt", "r");
  int n;

  for (n = 0; fscanf(f, "%s %s %d", l->fname, l->lname, &l->id)!=EOF; n++, l++)
    ;

  fclose(f);

  return n;
}
```

ب) رابطه کوچکتر و بزرگتر را برای دو ساختار **Student** به این صورت تعریف می کنیم: در ابتدا نام خانوادگی را مبنا قرار می دهیم. اگر نام خانوادگی دو دانشجو با هم متفاوت بود، دانشجویی بزرگتر است که نام خانوادگی او (به صورت آرایه ای از کاراکترها) بزرگتر باشد. اگر نام های خانوادگی مساوی بود، دانشجویی بزرگتر است که نام کوچک او (به صورت آرایه ای از کاراکترها) بزرگتر است. اگر نام های کوچک نیز مساوی بودند دانشجویی بزرگتر است که شماره دانشجویی آو (به عنوان یک **عدد صحیح**) بزرگتر است. شما باید تابع

```
int cmpStudent(struct Student st1, struct Student st2);
```
را طوری بنویسید که **st1** و **st2** را مقایسه کند. اگر **st1** از **st2** کوچکتر بود یک عدد منفی، اگر **st1** بزرگتر بود یک عدد مثبت و اگر **st1** و **st2** مساوی بودند عدد صفر را به عنوان مقدار بازگشتی برگرداند. برای مقایسه رشته ها می توانید از تابع استاندارد **strcmp** استفاده کنید:

```
int strcmp(char s1[], char s2[]);
```

این تابع دو رشته حرفی را می گیرد، اگر **s1** از **s2** بزرگتر، کوچکتر یا با آن مساوی بود به ترتیب یک عدد مثبت، یک عدد منفی و صفر بر می گرداند. **(۱۰ نمره، زمان ۱۰ دقیقه)**

```
int cmpStudent(struct Student st1, struct Student st2) {
  int cmp;

  if ((cmp = strcmp(s1.lname, s2.lname)) != 0)
    return cmp;

  if ((cmp = strcmp(s1.fname, s2.fname)) != 0)
    return cmp;

  return s1.id - s2.id;
}
```

ج) با استفاده از تابع بالا تابعی بنویسید که آرایه ای از ساختارهای **Student** را بگیرد و آنها را به ترتیب صعودی مرتب کند. **(۱۰ نمره، زمان ۱۰ دقیقه)**

```
void sortStudents(struct Student l[], int n) {
  for (int m = n-1; m > 1; m--)
    for (int i = 0; i < m; i++)
      if (cmpStudent(l[i], l[i+1]) > 0) {
        struct Student tmp = l[i];
        l[i] = l[i+1];
        l[i+1] = tmp;
      }
}
```

د) تابع **writeStudents** آرایه ای از ساختارهای Student را بعلاه طول آرایه به عنوان آرگومان می گیرد و با استفاده از تابع fwrite آن را به صورت دسترسی اتفاقی (بایت به بایت) در فایلی با نام studentList.bin ذخیره می کند. پروتوتایپ تابع fwrite به صورت زیر است:

```
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream)
```

بدنه تابع **writeStudents** را بنویسید. **(۵ نمره، زمان ۵ دقیقه)**

```
void  writeStudents(struct Student l[], int n) {
  FILE *f = fopen("studentlist.bin", "w");

  fwrite(l, sizeof(struct Student), n, f);

  fclose(f);

}
```

**سوال ۴-** فرض کنید اعداد مبنای ۱۶ را به صورت رشته C (آرایه ای از کاراکترها که با 0 پایان یافته اند) نشان می‌دهیم. بدین ترتیب که ارقام 0 تا 9 و A تا F با کد ASCII خود نمایش داده می شوند. همچنین برای راحتی کار شما فرض می کنیم رقم یکان در خانه صفر آرایه، رقم شانزده گان در خانه ۱ آرایه و... قرار داده می شود. بنابراین اعداد به صورت برعکس نوشته می شوند. برای مثال دستورات

```
char a[] = "9CA2";
char b[] = "ABCDF8";
```

قرار است دو عدد 2AC9 و 8FDCBA در مبنای ۱۶ را نشان دهند. می خواهیم تابعی به نام add بنویسیم که دو عدد از نوع بالا را با هم جمع کند و در آرایه سومی قرار دهد. در کد زیر مثالی از استفاده از تابع add را می بینید:

```
int main() {
  char a[] = "9CA2";
  char b[] = "ABCDF8";
  char c[100];

  add(c,a,b);

  puts(a);
  puts(b);
  puts(c);
}
```

شما باید بدنه تابع add را بنویسید. **(۲۵ نمره، زمان ۳۰ دقیقه)**

**۵ نمره اضافی** برای کسانی منظور خواهد شد که تابع add را **صرفا با استفاده از اشاره گر** و بدون استفاده از عملگر براکت [ ] بنویسند.

```
void add(char *c, char *a, char *b) {
  int carry = 0;
  int i;

  for (i = 0; a[i] != 0 && b[i] != 0; i++) {
    int da = a[i] >= '0' && a[i] <= '9' ? a[i] - '0' : a[i] - 'A' + 10;
    int db = b[i] >= '0' && b[i] <= '9' ? b[i] - '0' : b[i] - 'A' + 10;
    int dc = da + db + carry;

    carry = dc / 16;
    dc    = dc % 16;

    c[i] = dc < 10 ? '0' + dc : 'A' + dc - 10;
  }

  if (a[i] == 0)
    a = b;

  for (; a[i] != 0 ; i++) {
    int da = a[i] >= '0' && a[i] <= '9' ? a[i] - '0' : a[i] - 'A' + 10;
    int dc = da + carry;

    carry = dc / 16;
    dc    = dc % 16;

    c[i] = dc < 10 ? '0' + dc : 'A' + dc - 10;
  }

  if (carry)
    c[i++] = '1';

  c[i] = 0;
}
```

**سوال۵-** تابعی بنویسید که یک آرایه دو بعدی با M سطر و N ستون را به عنوان آرگومان بگیرد و اعداد 1 تا M*N را به صورت مارپیچ (مانند شکل زیر) در آن ذخیره کند. مثال زیر برای M=5 و N=4 نوشته شده است. فرض کنید M و N بوسیله define تعریف شده اند. **(۱۵ نمره، زمان ۲۰ دقیقه)**

| | | | |
|---|---|---|---|
| 4 | 3 | 2 | 1 |
| 5 | 16 | 15 | 14 |
| 6 | 17 | 20 | 13 |
| 7 | 18 | 19 | 12 |
| 8 | 9 | 10 | 11 |

```
#define M 7363
#define N 54221
```

```
void spiral(int A[M][N]) {
  //          {left,down,right, up}
  int di[] =   {0,  1,   0,   -1};
  int dj[] =   {-1, 0,   1,    0};
  int endi[] = {0,  M-1, M-1, 1};
  int endj[] = {0,  0,   N-1, N-1};

  int counter = 1;

  int i = 0;
  int j = N-1;
  int k = 0;
  while (counter <= M*N) {
    A[i][j] = counter++;

    if (i == endi[k] && j == endj[k]) {
      int next_k = (k+1)%4;
      endi[k] -= di[k];
      endj[k] -= dj[k];
      endi[k] += di[next_k];
      endj[k] += dj[next_k];
      k = next_k;
    }

    i += di[k];
    j += dj[k];
  }
}
```