درس برنامه نویسی مقدماتی

استاد درس: آقای دکتر پورفرد تدریسیار: آقای دکتر قنبرپور

اعضای گروه:

- سرگروه: نادیا غلامی
- نگار کاظمی

توضیح کد پروژه‌ی پایانی

۱) در ابتدا برای شروع پروژه ساختاری به نام Student درست کردیم و با ساخت یک آرایه صدتایی از جنس این ساختار با اطلاعات صد دانشجو را درون فایل projectlist.txt ذخیره کردیم.

این بخش پروژه توسط نگار کاظمی انجام گرفت.

۲)خواندن فایل ذخیره اطلاعات دانشجویان در لیست پیوندی

سپس برای ایجاد یک لیست پیوندی دو طرفه از اطلاعات دانشجویان یک ساختار با نام Node و یک پوینتر prev، یک پوینتر به نام Student ازجنس ساختار dataساختیم که در آن یک متغیر به نام ایجاد کردیم. و یک آرایه صد و یک از این ساختار درست کردیم. برای ایجاد لیست پیوندی nextبه نام و Student استفاده کردیم. این تابع دو ورودیبه ترتیب از جنس های ساختار insertمورد نظر از تابع تعریف کردیم و با استفاده از دستور Node می گیرد. در این تابع سه پوینتر از جنس ساختار Node به نود جدید حافظه اختصاص دادیم.malloc

```
187 void insert(struct Node** head, struct Student student)
188 {
189
190     //dadan hafeze
191     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
192     newNode->data = student;
193     newNode->next = NULL;
194
195     if (*head == NULL) {
196         newNode->prev = NULL;
197         *head = newNode;
198     } else {
199         struct Node* lastNode = *head;
200         while (lastNode->next != NULL)
201         {
202             lastNode = lastNode->next;
203         }
204         lastNode->next = newNode;
205         newNode->prev = lastNode;
206     }
207 }
208
```

ها به صورت یک node تا زمانی که به انتهای لیست برسیم whileسپس با استفاده از حلقه قرار دادیم. و سپس در تابع NULL نود آخر را برابر با node nextلیست بهم متصل می شوند و در انتها اول را node برای درست کردن جدولی از اطلاعات دانشجویان ابتدا مشخصات printLinkedList نشده باشد NULL مورد نظر برابر با node تا مانی کwhileچاپ کردیم و سپس با استفاده از حلقه بعدی آن قرار دادیم. بدین ترتیب node مورد نظر را برابر با node را چاپ کردیم و nodeاطلاعات آن لیستی از مشخصات دانشجویان در قالب لیست پیوندی دوطرفه در ترمینال چاپ شد.

```
210 void printLinkedList(struct Node* head)
211 {
212     printf("%-5s%-20s%-20s%-20s%-20s\n","row" ,"First Name", "Last Name", "ID", "City");
213
214     struct Node* CurrentNode = head;
215     int i = 1;
216
217     while (CurrentNode != NULL)
218     {
219         printf("%-5d%-20s%-20s%-20s%-20s\n",
220                i,
221                CurrentNode->data.FirstName,
222                CurrentNode->data.LastName,
223                CurrentNode->data.ID,
224                CurrentNode->data.City);
225
226         CurrentNode = CurrentNode->next;
227         i++;
228     }
229 }
230
```

این بخش توسط دو عضو گروه انجام گرفت.

درصد مشارکت:

نگار کاظمی ۵۰ درصد

نادیا غلامی ۵۰ درصد

۳) جست‌وجو

ابتدا یک متغییر به نام targetRow تعریف کردیم و از کاربر خواستیم که ردیف موردنظرش را انتخاب کند. سپس در ردیف اول قرار گرفتیم و پوینتری از جنس نود ساختیم و در اولین نود قرار دادیم و تا زمانی که به نود نال برسیم بررسی کردیم که آیا ردیفی که در آن قرار داریم برابر با ردیف مورد نظر کاربر هست یا خیر اگر بود مشخصات دانشجو در آن ردیف را با کمک لیست پیوندی چاپ می کردیم و در اگر نبود یک نود و یک ردیف جلو می رفتیم.

این بخش توسط نادیا غلامی انجام گرفت.

```c
void search(struct Node** head)
{

    int targetRow;

    puts("enter the row you want");
    scanf("%d", &targetRow);

    int currentRow = 1;
    struct Node* current = *head;

    while (current != NULL)
    {
        if (currentRow == targetRow)
        {

            printf("founded!!!!!!!!! row %d:\n", currentRow);
            printf("hhhhhhh");

            printf("First Name: %s\n", current->data.FirstName);
            printf("Last Name: %s\n", current->data.LastName);
            printf("ID: %s\n", current->data.ID);
            printf("City: %s\n", current->data.City);
            return;
        }

        // bro badi
        current = current->next;
        currentRow++;
    }

    printf("No match found!!!!!!!!\n");
}
```

۴) اضافه کردن دانشجو

در این بخش ابتدا از کاربر شماره ردیف مورد نظر، نام و نام خانوادگی، شماره دانشجویی و محل تولد او را از کاربر دریافت کرده و در یک node جدید ریختیم. سپس با استفاده از تابع lowercase حروف نام و نام خانوادگی وارد شده توسط کاربر را به حروف کوچک تبدیل کرده و در همان متغییرها ذخیره می کنیم.

در مرحله بعد برای عددی که کاربر وارد می کند سه حالت را درنظر گرفتیم.

حالت اول: در این حالت اگر کاربر عددی کمتر از یک وارد کند، ما درنظر گرفتیم که می خواهد دانشجو به اول لیست اضافه شود و اطلاعات این دانشجو را درون data یک node جدید ریختیم و next این node را با prev اولین خانه node قبلی خود، prev اولین خانه node قبلی رابا next این node و prev این node جدید را با NULL برابر قرار دادیم. در قسمت بعد مانند تابع printLinkedList با استفاده از حلقه جدولی از مشخصات دانشجویان به همراه دانشجوی اضافه شده در ترمینال چاپ کردیم.

حالت دوم: در این حالت اگر کاربر عددی بزرگتر از صد وارد کند، ما درنظر گرفتیم که می خواهد دانشجو به انتهای لیست اضافه شود و دوباره مانند حالت قبل اطلاعات را درون node جدید ذخیره کردیم با این تفاوت که این بار prev این node با next آخرین خانه node، next آخرین خانه node را با prev این node و next این nodeجدید را با NULL برابر قرار دادیم. سپس جدولی از مشخصات دانشجویان به همراه دانشجوی اضافه ده در ترمینال چاپ کردیم.

حالت سوم: در این حالت اگر کاربر عددی بین یک تا صد وارد کند. مانند دو حالت قبل اطلاعات دانشجو را درون newnode ذخیره کردیم حال اگر بخواهیم بین خانه اول تا اخر خونه اضافه کنیم باید پیوند بین دوتا خونه رو شکسته شود به طوری که اگر سر خانه اول به ته خانه دوم وصل باشد و بخواهیم بین این دوپخانه چیزی اضافه کنیم باید یک خانه اضافه شود پس خانه دوم در واقع تبدیل به خانه سومو می شود و حال خانه دوم انتهای خانه اول را به سر خانه دوم وصل می کند و انتهای خانه دوم را به سر خانه سوم. پس نوشتیم next خانه اول برابره با خانه دوم و پریو خانه دوم برابره با next خانه اول next خانه دوم برابر با خانه سوم و prev خانه سوم برابراست با خانه دوم.

این بخش توسط دو عضو گروه انجام گرفت.

درصد مشارکت:

نگار کاظمی ۵۰ درصد

نادیا غلامی ۵۰ درصد

```c
void addstudent(struct Node** head)
{

    int row;
    puts("enter row:");
    scanf("%d", &row);

    struct Student newStudent;

    puts("enter first name:");
    scanf("%s", newStudent.FirstName);

    puts("enter last name:");
    scanf("%s", newStudent.LastName);

    puts("enter city name:");
    scanf("%s", newStudent.City);

    puts("enter ID");
    scanf("%s", newStudent.ID);

    lowercase(newStudent);

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = newStudent;
    newNode->next = NULL;

    if (row <= 1)
    {
        newNode->prev = NULL;
        newNode->next = *head;
        if (*head != NULL)
        {
            (*head)->prev = newNode;
        }
        *head = newNode;
    }
```

```c
else
{
    struct Node* current = *head;
    int i = 1;
   while (current != NULL && current->next != NULL)
    {
        current = current->next;
        i++;
    }

    if (row < i)
    {
        //specified row
        current = *head;
        i = 1;
        while (current != NULL && i < row - 1)
        {
            current = current->next;
            i++;
        }

        newNode->next = current->next;
        newNode->prev = current;
        if (current->next != NULL)
        {
            current->next->prev = newNode;
        }
        current->next = newNode;
    }
    else
    {
        //end of the list
        newNode->prev = current;
        current->next = newNode;
    }
}
```

۵) حذف دانشجو

در این بخش ابتدا از کار ردیف مورد نظرش را دریافت کردیم و در targetRow ذخیره کردیم سپس از اولین ردیف و اولین Node تا زمانی که نود ما مخالف NULL باشد شروع به بررسی کردیم و وقتی به آن Node (current)رسیدیم که ردیف آن با ردیف مورد نظر کاربر برابر بود آنگاه next نود قبل آن را (PrevCur) برابر با نود بعدی آن(AfterCur) و prev نود بعد آن را با نود قبلی برابر قرار دادیم. بدین ترتیب دانشجوی موردنظر از لیست پیوندی حرف شد و لیست را به صورت جدول همانطور که در تابع printLinkedList اشاره کردیم در ترمینال چاپ کردیم.

این بخش توسط هر دو عضو گروه انجام شد.

درصد مشارکت:

نگار کاظمی ۵۰ درصد

نادیا غلامی ۵۰ درصد

```
    int targetRow;

    puts("enter the row you want to delelte:");
    scanf("%d", &targetRow);

    int currentRow = 1;

    struct Node* PrevCur = *head;
    struct Node* current;
    current = PrevCur->next;
    struct Node* AfterCur;
    AfterCur = current->next;

    while (current != NULL)
    {

        if (currentRow == targetRow)
        {
            PrevCur->next = AfterCur;
            AfterCur->prev = PrevCur;
        }

        current = current->next;
        currentRow++;
    }


    printf("%-5s%-20s%-20s%-20s%-20s\n","row" ,"First Name", "Last Name", "ID", "City");

    struct Node* CurrentNode = *head;
    int i = 1;

    while (CurrentNode != NULL)
    {
        printf("%-5d%-20s%-20s%-20s%-20s\n",
               i,
               CurrentNode->data.FirstName,
               CurrentNode->data.LastName,
               CurrentNode->data.ID,
               CurrentNode->data.City);

        CurrentNode = CurrentNode->next;
        i++;
    }
```

۶) به هم زدن تصادفی

در این بخش با استفاده از کتابخانه‌های time.h و stdlib.h تابعی با نام randomsort تعریف کردیم. در این تابع ابتدا یک متغییر از جنس ساختار Studentبه نام temp تعریف کردیم. سپس متغییر j را تعریف کردیم و در یک حلقه for صد مرتبه یک عدد تصادفی بین صفر تا صد و کمتر با استفاده از تابع rand تولید کرده و در j می ریزیم و جای خانه iام(متغییر حلقه) Students را با خانه ی زام آن ([j]Students) با کمک temp عوض می کنیم. بدین ترتیب به صورت تصادفی و بدون ایجاد تکرار شماره ردیف دانشجویان بهم می ریزد. سپس در یک حلقه دیگر به ترتیب لیست را به صورت جدول در ترمینال چاپ کردیم. سپس یک فایل با نام randomlist با مود write ایجاد کردیم واین لیست تصادفی ای که درست کردیم را در نوشتیم.

این بخش توسط نگار کاظمی انجام شد.

```c
void randomsort(struct Student Students[])
{

    srand(time(NULL));
    int size = 100;
    struct Student temp;
    for (int i = size-1; i>0; i--)
    {
        int j = rand()%(i+1);
        temp = Students[i];
        Students[i] = Students[j];
        Students[j] = temp;
    }
    printf("randoooom\n");

     for(int i = 0; i<size; i++)
        printf("%-5d%-20s%-20s%-20s%-20s\n", i+1 ,Students[i].FirstName, Students[i].LastName, Students[i].ID, Students[i].City);

    FILE *pfile2;
    pfile2=fopen("RandomList.txt","w");
    if (pfile2==NULL)
        printf("Error opening file\n");
    fprintf(pfile2,"row  First Name          Last Name           ID                Place of birth\n");
    for(int i=0; i<100; i++)
        fprintf(pfile2,"%-5d%-20s%-20s%-20s%-20s\n", i+1 ,Students[i].FirstName, Students[i].LastName, Students[i].ID, Students[i].City);

    fclose(pfile2);

}
```

۷)مرتب سازی

در این بخش ابتدا در تابع sort اول مشخصه موردنظر و نوع مرتب سازی و صعودی یا نزولی بودن آن را از کاربر می گیریم سپس با توجه به ورودی های کاربر وارد تابع های موردنیاز را نوشتیم.

- Selectionsort: در این نوع مرتب سازی اگربرای مثال کاربر قصد مرتب کردن نام های کوچک دانشجویان به صورت صعودی را داشته باشد. ابتدا یک متغییر به نام temp از نوع ساختار Studentبرای جابه جا کردن دانشجویان ایجاد کردیم سپس در حلقه ای از صفر تا نود و نه

سپس در گام اول کمترین مقدار را برای اولین خانه در نظر گرفتیم و درحلقه ای دیگر از مقدار خانه بعدی با کمک تابع strcmp حروف اول هر دو اسم را باهم مقایسه کردیم اگر حرف اول اسم اول بزرگتر باشد تابع عدد منفی یک را برمیگرداند. در این صورت اندیس کمترین مقدار را برابر با اندیس اسم دوم قرار دادیم و سپس اسم اول را با اسمی که اندیس آن کمترین مقدار است جا به جا کردیم. به این شکل اولین اسم لیست کوچکترین حرف اول را دارد و این حلقه آنقدر تکرار می‌شود تا اسامی به صورت صعودی مرتب شوند. در بقیه موارد هم به این شکل هر سری بین اعداد کوچکترین عدد را می‌گیرد و در ابتدای لیست قرار می‌دهد و در آخر لیست به طور کامل شکل گیرد.

```
void selectionsort(struct Student Students[], int how, int which )
{

   printf("selection sort\n");
   long tick1= clock();

   if(which == 1)
   {
       //soudi
       if( how == 1)
       {
           int i, j, min_idx;
           int size = 100;
           struct Student temp;
           for (i = 0; i < size - 1; i++)
           {
               min_idx = i;
               for (j = i + 1; j < size; j++)
               {
                   if (strcmp(Students[j].FirstName, Students[min_idx].FirstName) < 0)
                   {
                       min_idx = j;
                   }
               }
           temp = Students[min_idx];
           Students[min_idx] = Students[i];
           Students[i] = temp;
           }
       }
```

- insertionsort : در این نوع مرتب سازی اگر برای مثال کاربر قصد مرتب کردن نام های کوچک دانشجویان به صورت صعودی را داشته باشد با استفاده از یک حلقه for در آن یک key تعریف کرده و آن را برابر با اطلاعات دومین دانشجو قرار دادیم و هر بار نام دانشجویی که در key قرار دارد را با نام دانشجوهای قبل آن را با استفاده از تابع srtcmp مقایسه کردیم و اگر حرف آن بزرگ تر بود و خروجی تابع مساوی یک شد. سراغ دانشجوی بعدی میرویم و اگر حرف اول آن کوچتکر بود دوباره با همه ی دانشجوهای قبل خود مقایسه می شود تا در جایگاه خود قرار گیرد. بدین ترتیب با تکرار حلقه‌ها اسامی به صورت صعودی مرتب می شوند.

```
697   void insertionsort(struct Student Students[], int how, int which )
698   {
699
700       printf("insertion sort\n");
701       int i, j, size = 100;
702       struct Student key;
703       long tick1 = clock();
704       // name
705       if(which == 1)
706       {
707
708           //soudi
709           if(how == 1)
710           {
711               for (i = 1; i < size; i++)
712               {
713
714                   key = Students[i];
715                   j = i - 1;
716                   while (j >= 0 && strcmp(key.FirstName, Students[j].FirstName)>0)
717                   {
718                       Students[j + 1] = Students[j];
719                       j = j - 1;
720                   }
721                   Students[j +1] = key;
722
723               }
724
725           }
```

- Bubblesort : در این نوع مرتب سازی نیز اگر برای مثال کاربر قصد مرتب کردن نام‌های کوچک دانشجویان را داشته باشد پس به دو حلقه فور نیاز داریم که ابتدا نام دانشجوی اول را با همه‌ی دانشجویان چک می‌کند و اگر حرف اول اسم این دانشجو از حرف اول هر دانشجوی دیگری کوچکتر بود با کمک متغییر temp از نوع ساختار student جا به جا می‌شود و بدین ترتیب با تکرار حلقه‌ها این لیست به صورت صعودی مرتب می‌شود.

```
void bubblesort(struct Student Students[], int how, int which )
{


    struct Student temp;
    int size = 100;
    printf("buble sort\n");

    long tick1= clock();

    // soudi
    if (which == 1)
    {

        if(how == 1)
        {

            for (int i = 0; i < size; i++)
            {
                for(int j = 0; j <size; j++)
                {

                    if(strcmp(Students[j].FirstName,Students[i].FirstName)>0)
                    {
                        temp = Students[i];
                        Students[i] = Students[j];
                        Students[j] = temp;

                    }
                }
            }

        }
```

این بخش توسط هر دو عضو گروه انجام شد.

در هرکدام از این بخش ها با استفاده از تابع clock زمان اولیه و انتهای مرتب سازی را مشخص کردیم و تفاضل این دو تابع را برابر با زمان اجرای برنامه قرار دادیم و برحسب ثانیه محاسبه کردیم. در ترمینال چاپ کردیم.

درصد مشارکت:

نگار کاظمی ۵۰ درصد

نادیا غلامی ۵۰ درصد

۸) نوشتن

در این مرحله تابعی با نام finalfile تعریف کردیم. ابتدا فایل اصلی خود (projectlist.txt) را با مود append باز کردیم و در آن لیست پیوندی رو پس از تغییراتی که روی آن صورت گرفت روی فایل اصلی نوشتیم.

سپس یک فایل باینری با نام binary.bin ایجاد کردیم و در آن با استفاده از دستور fwrite اطلاعات دانشجویان را در قالب لیست پیوندی پس از تغییرات صورت گرفته بر روی آن در فایل مورد نظر نوشتیم.

این بخش پروژه توسط نگار کاظمی انجام گرفت.

```c
void finalfile(struct Node** head)
{

    FILE *pfile;
    pfile=fopen("projectlist.txt","a");
    if (pfile==NULL)
        printf("Error opening file\n");
    fputs("_____\n",pfile);
    fprintf(pfile,"row  First Name          Last Name           ID                  Place of birth\n");
    struct Node* current_node = *head;
    int i = 1;

    while (current_node != NULL)
    {
        fprintf(pfile,"%-5d%-20s%-20s%-20s%-20s\n",
                i,
                current_node->data.FirstName,
                current_node->data.LastName,
                current_node->data.ID,
                current_node->data.City);

        current_node = current_node->next;
        i++;
    }
    fclose(pfile);

    FILE *pfile6;
    pfile6=fopen("binarylist.bin","wb");
    if (pfile6==NULL)
        printf("Error opening file\n");
    struct Node* current_node1 = *head;
    i = 1;

    while (current_node != NULL)
    {
        fwrite(&(current_node1->data),sizeof(current_node1->data),1,pfile6);
        current_node = current_node->next;
        i++;
    }
        fclose(pfile6);
```

۹)پایان برنامه

در این بخش کاربر با انتخاب این گزینه در پنل کاربری اجرای برنامه را به پایان می رساند.

۱۰) پنل کاربری

برای درست کردن graphic menu مورد نظرمان با کمک از ویدیو فرستاده شده و وارد کردن کد مورد نظر برای درست کردن پنجره، اول اندازه پنجره ای که باید باز شود رو میدهیم سپس با درست کردن مستطیل هایی با کد داده شده در مکان های مورد نظر قرار می دهیم.(طول، عرض، مکان در محور ایکس و محور ایگرگ) سپس هشت بار این کار را تکرار می کنیم و در اخر با اوردن شرطی که اگه روی هر کدام از این مستطیل ها کلیک شود برنامه مورد نظرش اجرا شود.

این بخش توسط نادیا غلامی انجام شد.