ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ

КУРСОВА ЗАДАЧА

ПО ПИК II

Разработил: *Константин Костов*

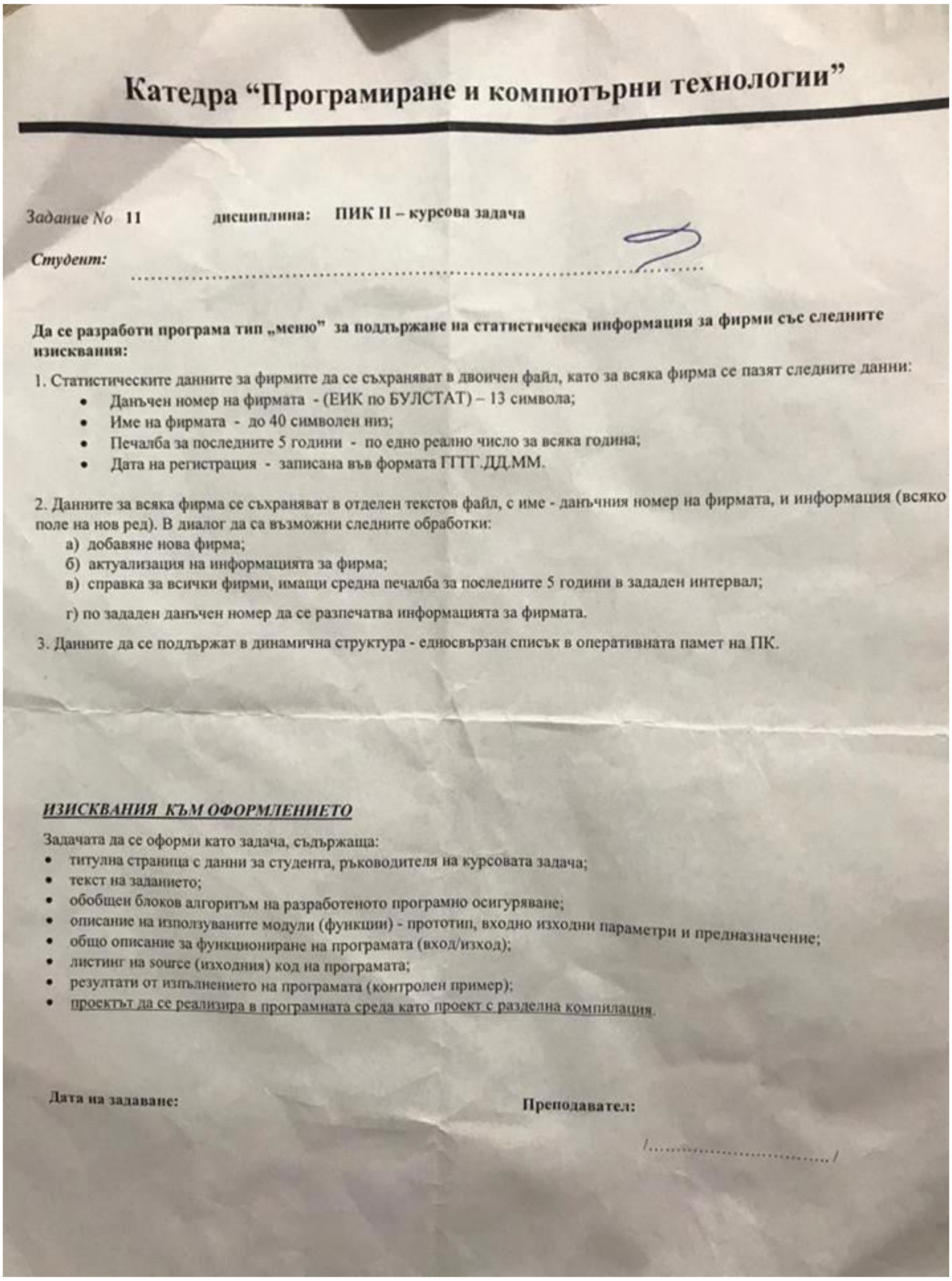
Ръководител:

Факултет: ФКСУ

Фак. №: 121217148

Дата:24.05.2017г.

1.Текст на заданието


Катедра "Програмиране и компютърни технологии"

Задание No 11 дисциплина: ПИК II – курсова задача

Студент: ....

Да се разработи програма тип „меню" за поддържане на статистическа информация за фирми със следните изисквания:

1. Статистическите данните за фирмите да се съхраняват в двоичен файл, като за всяка фирма се пазят следните данни:
   - Данъчен номер на фирмата - (ЕИК по БУЛСТАТ) – 13 символа;
   - Име на фирмата - до 40 символен низ;
   - Печалба за последните 5 години - по едно реално число за всяка година;
   - Дата на регистрация - записана във формата ГГГГ.ДД.ММ.

2. Данните за всяка фирма се съхраняват в отделен текстов файл, с име - данъчния номер на фирмата, и информация (всяко поле на нов ред). В диалог да са възможни следните обработки:
   а) добавяне нова фирма;
   б) актуализация на информацията за фирма;
   в) справка за всички фирми, имащи средна печалба за последните 5 години в зададен интервал;
   г) по зададен данъчен номер да се разпечатва информацията за фирмата.

3. Данните да се поддържат в динамична структура - едносвързан списък в оперативната памет на ПК.

<u>ИЗИСКВАНИЯ КЪМ ОФОРМЛЕНИЕТО</u>

Задачата да се оформи като задача, съдържаща:
- титулна страница с данни за студента, ръководителя на курсовата задача;
- текст на заданието;
- обобщен блоков алгоритъм на разработеното програмно осигуряване;
- описание на използуваните модули (функции) - прототип, входно изходни параметри и предназначение;
- общо описание за функциониране на програмата (вход/изход);
- листинг на source (изходния) код на програмата;
- резултати от изпълнението на програмата (контролен пример);
- <u>проектът да се реализира в програмната среда като проект с разделна компилация.</u>

Дата на задаване: **Преподавател:**

/.................................../

2. Обобщен блоков алгоритъм на разработеното програмно осигуряване

- ## Описание на използуваните модули (функции) - прототип, входно изходни параметри и предназначение

  void insertFirm()-функция, която дава възможност на потребителя да въведе информацията за нова фирма и записва новата информация накрая на списъка

  void printListStdNumber()- функцията извежда на екран информацията за дадена фирма по въведен от потребителя номер

  void printList- ()**-** функцията извежда на екран информацията за всички фирми

  void check()- Справка на всички фирми имащи средна печалба в зададен интервал

  void update_info()- ъпдейтва дадена информация за дадена фирма

  retrieve() -взима всички запазени фирми от бинарен файл и създава нов лист

  removeStdnumber() -изтрива фирма по зададен номер

  void SaveInFile()-функцията записва всички фирми, които сме записали в бинарен файл

- ## Общо описание за функциониране на програмата (вход/изход)

Програмата изкарва на екран меню с 9 опции. При избиране на дадена опция от потребителя, програмата изпълнява условието й и връща на екран резултата от нея. При избиране на ,изход‘ се излиза от програмата.

- ## Листинг на source (изходния) код на програмата

```
#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<stdbool.h>
```

```
#define NAMELEN 30 //Maximum characters for each name

#define MAXNUMGRADES 5



// LINKED LIST ---------------------------------------------------------------------
------------------------------------------


/* A structure that contains a string that would be used for the firms's name,

an integer which would be used for the firms's =number

and an array of floating points which would be used for the student's marks,

an integer for the number of grades introduced for each firm,

and a pointer to the next node in the linked list

*/

struct node

{

  char name[NAMELEN];

  int firmNumber;

  int  salary[MAXNUMGRADES];

  int date_reg;

  struct node *next;

  float average_salary;

};
```

```
// Function to insert new firm in Linked list in alphabetical order
void insertFirm(struct node** head)
{
    char wantedName[NAMELEN]; //String to store the name of the new
firm when a new firm is introduced
    printf("Add firm name: \n");
    scanf("%s",wantedName);

    int wantedfirmNumber; // Integer to store the firm number of the new firm
when a new firm is introduced
    printf ("Add firm number: \n");
    scanf("%d",&wantedfirmNumber);

    int wanted_salary[5];



   //Allocating memory for a new firm node
   struct node* new_node = (struct node*) malloc(sizeof(struct node));

   //We copy the name and firm number parameters into the name and firm
number of the firm
   strcpy(new_node->name, wantedName);
   new_node->firmNumber = wantedfirmNumber;
    printf("Salary for the first year\n");
    scanf("%d",&wanted_salary[0]);
    printf("Salary for the second year\n");
    scanf("%d",&wanted_salary[1]);

    printf("Salary for the third year\n");

    scanf("%d",&wanted_salary[2]);

    printf("Salary for the fourth year\n");

    scanf("%d",&wanted_salary[3]);

      printf("Salary for the fifth year\n");

    scanf("%d",&wanted_salary[4]);



   new_node->salary[0]=wanted_salary[0];

   new_node->salary[1]=wanted_salary[1];

   new_node->salary[2]=wanted_salary[2];

   new_node->salary[3]=wanted_salary[3];

   new_node->salary[4]=wanted_salary[4];

 new_node->average_salary = (new_node->salary[0] + new_node->salary[1]
+ new_node->salary[2] + new_node->salary[3] + new_node->salary[4])/5;

    int dd,mm,yy;

   int date;

 printf("Enter date (yy.dd.mm) format: ");

   scanf("%d.%d.%d",&yy,&dd,&mm);

    /*adding dd,mm,yy into date*/

   /*an integer has 4 bytes and dd range is 1 to 31 , mm range is 1 to 12
which

    *can be stored in 1 byte, 1 byte and in rest of 2 bytes

    *we can store year.*/
```

```
     new_node->date_reg=0;

     new_node->date_reg   |= (dd&0xff); /*dd storing in byte 0*/

     new_node->date_reg   |= (mm&0xff)<<8; /*mm storing in byte 1*/

     new_node->date_reg   |= (yy&0xffff)<<16; /*yy storing in byte 2 and 3*/



    //For traversing through the list, use two nodes next to each other moving
them forward simultaneously

    //pointer to the previous firm in the linked list

    struct node* previous = (struct node*)malloc(sizeof(struct node));

    previous = NULL;

    //pointer to the current firm in the linked list

    struct node* current = (struct node*)malloc(sizeof(struct node));

    current = *head;


    //Looping through the firms until the name of the current is alphabetically
'higher' then the one we want to insert

    while (NULL != current && strcmp(wantedName, current->name) > 0)

    {

        previous = current;

        current = current->next;

    }


    if (NULL == previous)

    {   //Insert at beginning of linked list

        new_node->next = current;
        *head = new_node;
    }
    else
    {   //Insert between previous and current node
        previous->next = new_node;
        new_node->next = current;
    }
}




//Function to delete firm from the list using the firm's  number
void removeStdNumber(struct node **head){

  int wantedStudentNumber; //Integer to store the firm number of the firm
that the user wished to delete
  printf("Please enter Firm number: \n");
  scanf("%d",&wantedStudentNumber);

  // For traversing through the list, we use two nodes next to each other
moving them forward simultaneously
  // Use a third node as an auxiliary node when deleting the node in order to
fix the links
  struct node *current;
  current=*head;
```

```
struct node *previous;
previous=NULL;
struct node *temp;
temp=NULL;

// if list is empty
if(current == NULL){
  printf("List empty, no items to delete\n\n");
  return;
}

while(current != NULL){

  // If match found
  if(wantedStudentNumber == current->firmNumber){

            // Delete at the start of the list
            if (current == *head){
                     temp=current;
                     *head=current->next;

                     free(temp);
                     printf("Firm has been removed successfully!\n\n");
            }
            else{
                temp=current;

                previous->next=current->next;


                free(temp);

                printf("Firm has been removed successfully!\n\n");

            }

            return;

       }

     // If match not found move on to next

     previous=current;

     current = current->next;

    }

    //No match found, so error

    printf("Student has not been introduced yet! \n\n");

}



// Function to update infor of a firm

void update_info(struct node **head){


  char wantedName[NAMELEN]; //Integer to store the name of the firm
whose report we wish to print

  printf("Please enter firm name: \n");

  scanf("%s",wantedName);
```

```
char wantedName1[NAMELEN]; //String to store the name of the new firm
when a new student is introduced


  // Node used for traversing the list

  struct node *temp;

  temp=*head;

  while(temp !=NULL){

    // If match found

    if (strcmp(wantedName,temp->name)==0){

                char wantedName1[NAMELEN]; //String to store the name of the
new firm when a new student is introduced

     printf("Add firm name: \n");

     scanf("%s",wantedName1);


     int wantedfirmNumber; // Integer to store the firm number of the new firm
when a new firm is introduced

     printf ("Add firm number: \n");

     scanf("%d",&wantedfirmNumber);

    strcpy(temp->name, wantedName1);

    temp->firmNumber = wantedfirmNumber;

     int wanted_salary[5];

printf("Salary for the first year\n");

     scanf("%d",&wanted_salary[0]);

     printf("Salary for the second year\n");

     scanf("%d",&wanted_salary[1]);

     printf("Salary for the third year\n");
     scanf("%d",&wanted_salary[2]);

     printf("Salary for the fourth year\n");

     scanf("%d",&wanted_salary[3]);

       printf("Salary for the fifth year\n");

     scanf("%d",&wanted_salary[4]);



    temp->salary[0]=wanted_salary[0];

    temp->salary[1]=wanted_salary[1];

    temp->salary[2]=wanted_salary[2];

    temp->salary[3]=wanted_salary[3];

    temp->salary[4]=wanted_salary[4];

     int dd,mm,yy;

    int date;

 printf("Enter date (yy.dd.mm) format: ");

    scanf("%d.%d.%d",&yy,&dd,&mm);

     /*adding dd,mm,yy into date*/

    /*an integer has 4 bytes and dd range is 1 to 31 , mm range is 1 to 12
which

     *can be stored in 1 byte, 1 byte and in rest of 2 bytes

     *we can store year.*/


     temp->date_reg=0;

     temp->date_reg   |= (dd&0xff); /*dd storing in byte 0*/

     temp->date_reg   |= (mm&0xff)<<8; /*mm storing in byte 1*/
    temp->date_reg   |= (yy&0xffff)<<16; /*yy storing in byte 2 and 3*/







    }

   // If not a match, move to next item

     temp = temp->next;

  }

  //No match found, so error

  printf("Firm has not been introduced yet, please introduce student first in
order to print their report\n");

}

// function to : printf(" 9) Spravka na vsichki firmi imashti sredna pechalba v
zadaden interval\n\n");

void check(struct node **head){

    printf("Spravka na vsichki firmi imashti sredna pechalba v zadaden
interval\n");

    printf("print the first digit\n");

    float digit1;

    float digit2;

    scanf("%f",&digit1);

    printf("print the second digit\n");

    scanf("%f",&digit2);


    struct node *temp;
  temp=*head;

  int dd;

  int mm;

  int yy;

  if (*head == NULL){

    printf("No Firms have been introduced!\n");

   }

   while(temp != NULL) {

     if(temp->average_salary >digit1 && temp->average_salary<digit2){


        printf("Name: %s\n",temp->name);

}


    temp = temp->next;

    }








}
```

```
// Function to print report for one firm using firm number
void printListStdNumber(struct node **head){


    int wantedFirmNumber; // Integer to store firm number of firm whose
report the user wihes to print
  printf("Please enter firm number: \n");
  scanf("%d",&wantedFirmNumber);


  struct node *temp;
  temp=*head;
  while(temp !=NULL){
    // If match found
    if (wantedFirmNumber==temp->firmNumber){
      printf("Name: %s\n",temp->name);
      printf("Firm number: %d\n",temp->firmNumber);
      printf("Salaries: \n");
      for(int i=0; i<5;i++){
              printf("\t Salary no.%d: %.2d\n",i+1,temp->salary[i]);
      }
      return;
    }
    // If not a amatch, move to next item
    temp = temp->next;
  }
```

```
    //No match found, so error
    printf("Firm has not been introduced yet, please introduce student first in
order to print their report\n");
}


// Function to print report of all firms


void printList(struct node **head) {
  struct node *temp;
  temp=*head;
  int dd;
  int mm;
  int yy;
  if (*head == NULL){
    printf("No Firms have been introduced!\n");
   }
   while(temp != NULL) {
    printf("Name: %s\n",temp->name);
    printf("Firm number: %d\n",temp->firmNumber);
    printf("Salaries: \n");
      for (int i=0; i<5; i++){
        printf("\t Salary no.%d: %.2d\n",i+1,temp->salary[i]);
      }
 dd = (temp->date_reg &0xff); /*dd from byte 0*/
```

```
     mm = ((temp->date_reg>>8)& 0xff); /*mm from byte 1*/
     yy = ((temp->date_reg>>16)&0xffff); /*yy from byte 2 and 3*/

     printf(" Date: %04d.%02d.%02d\n",yy,dd,mm);
      printf("\n\n");

    temp = temp->next;
   }
}

// Function to save data to file
void saveToFile(struct node **head){

  FILE *outfile;
  outfile = fopen ("Firms", "wb");
  struct node *temp;
  temp=*head;

  if (outfile == NULL){
    printf("Error opening file\n");
    return;
    }
  while(temp != NULL){
     fwrite(temp,sizeof(struct node),1,outfile);
    temp=temp->next;
  }
  printf("Data saved successfully!\n\n");
  fclose(outfile);
}

// Function to retrieve data from file
void retrieve(struct node **head){
    struct node *tempObject = (struct node*)malloc(sizeof(struct node));
    struct node *ptr;
    struct node *previous;
    FILE *file=fopen("Firms", "rb");
    *head=tempObject;
    if (file != NULL){
        do{
                    fread(tempObject,sizeof(struct node),1, file);
            ptr=*head;
            previous=*head;
            while(previous->next != NULL){
                ptr= (struct node *)malloc(sizeof(struct node));
                fread(ptr, sizeof(struct node), 1,file);
                previous->next=ptr;
                previous=ptr;
            }
        }while(fread(tempObject, sizeof(struct node), 1,file) == 1);
    }
    else{
      printf("Error opening file\n");
      return;
    }
    printf("Data retrieved successfully!\n\n");
}

//------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

int main(int argc, char const *argv[]){
  // Boundary condition so that the program does not crash

  int menu_choice;
  int menu_choice1;
  int menu_choice2;
```

```
 int menu_choice3;
 int menu_choice4;
 int menu_choice5;
 int menu_choice6;
 int menu_choice7;
 int menu_choice8;
 int menu_choice9;
 struct node *head=NULL;


 do{
   // Print menu on screen
   printf ("  Menu:\n\n");
   printf (" 1) Introduce Firm \n");
   printf (" 2) Remove firm \n");


   printf (" 3) edit information \n");
   printf (" 4) Print report for a firm\n");
   printf (" 5) Print report for all firms\n");
   printf (" 6) Save to File\n");


   printf (" 7) Retrive from FIle\n");
printf (" 8) Exit\n");
printf(" 9) Spravka na vsichki firmi imashti sredna pechalba v zadaden
interval\n\n");
```

```
    printf (" Please select an option from the menu: \n");
    scanf ("%d" , &menu_choice); // Read user input for menu choice

    switch (menu_choice)
    {
     case 1:
       printf("Press 1 if you wish to continue:\n");
       printf("Press 0 to go back to Menu :\n");
       scanf("%d",&menu_choice1);

       if (menu_choice1==1){
         insertFirm(&head);
        }

       else if(menu_choice1==0){
         break;
       }

       else{
         printf("Invalid choice, please select your option from the menu once
again:\n");
       }

       break;
```

```
case 2:

  printf("Press 1 to remove Firm using their number:\n");


  printf("Press 0 to go back to Menu:\n");

  scanf("%d",&menu_choice2);


   if (menu_choice2==1){

    removeStdNumber(&head);

   }


  else if(menu_choice2==0){

    break;


  }

  else{

    printf("Invalid choice, please select your option from the menu once
again:\n");

  }

  break;


case 3:


  printf("Press 1 to update information about the firm:\n");

  printf("Press 0 to go back to Menu:\n");

  scanf("%d",&menu_choice3);
```

```
        if (menu_choice3==1){
         update_info(&head);
        }




        else if(menu_choice3==0){
          break;
        }

        else{
          printf("Invalid choice, please select your option from the menu once
again:\n");
        }

        break;


      case 4:
        printf("Press 1 to print report for an individual firm using the firms's
number:\n");

        printf("Press 0 to go back to Menu:\n");
        scanf("%d",&menu_choice4);
```

```
    if (menu_choice4==1){
      printListStdNumber(&head);
    }

    else if(menu_choice4==0){
      break;
    }

    else{
      printf("Invalid choice, please select your option from the menu once
again:\n");
    }
    break;

   case 5:
    printf("Press 1 if you wish to continue:\n");
    printf("Press 0 to go back to Menu :\n");
    scanf("%d",&menu_choice5);

    if (menu_choice5==1){
      printList(&head);
     }
```

```
        else if(menu_choice5==0){
          break;
        }

        else{
          printf("Invalid choice, please select your option from the menu once
again:\n");
        }

        break;

      case 6:
        printf("Press 1 if you wish to continue:\n");
        printf("Press 0 to go back to Menu :\n");
        scanf("%d",&menu_choice6);

        if (menu_choice6==1){
          saveToFile(&head);
         }

        else if(menu_choice6==0){
          break;
        }
        else{
          printf("Invalid choice, please select your option from the menu once
again:\n");
        }

        break;

         case 7:
        printf("Press 1 if you wish to continue:\n");
        printf("Press 0 to go back to Menu :\n");
        scanf("%d",&menu_choice7);

        if (menu_choice7==1){
          retrieve(&head);
         }

        else if(menu_choice7==0){
          break;
        }

        else{
          printf("Invalid choice, please select your option from the menu once
again:\n");
        }

           break;
```

```
     case 8:
      printf("Press 1 if you wish to continue:\n");
      printf("Press 0 to go back to Menu :\n");
      scanf("%d",&menu_choice8);

      if (menu_choice8==1){
        printf ("You have selected to exit the menu \n");
       }

      else if(menu_choice8==0){
        break;
      }



      else{
        printf("Invalid choice, please select your option from the menu once
again:\n");
      }

      break;
         case 9:
      printf("Press 1 if you wish to continue:\n");
      printf("Press 0 to go back to Menu :\n");
      scanf("%d",&menu_choice8);
```

```
        if (menu_choice9==1){
          check(&head);
         }

        else if(menu_choice9==0){
          break;
        }



        else{
          printf("Invalid choice, please select your option from the menu once
again:\n");
        }

        break;




      default:
        printf("Invalid choice, please select your option from the menu once
again\n");
        break;

  }
}while (menu_choice!=8 || menu_choice8!=1);
```

}