**Question # 1               [40 points (4 each), 70 mins] CLO1**

|  |  |
| --- | --- |
| Considering the following programs and illustrate the required process in graphical form. Assume all necessary header files are included and all programs are syntactically correct. | |
| 1. **Illustrate a memory allocation for both type of dynamic memory allocation**.   void main()   {  double \*ptr1,\*ptr2;  ptr1=(double\*)malloc(5 \* sizeof(double));  ptr2=(double\*)calloc(5 , sizeof(double));}    Show dummy addresses and garbage values to highlight the difference.  **Note: GB stands for Grabbage Value**   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Variable** | **ptr1+0** | **ptr1+1** | **ptr1+2** | **ptr1+3** | **ptr1+4** | | Address | 1000 | 1008 | 1016 | 1024 | 1032 | | Value | GB | GB | GB | GB | GB | | **Variable** | **ptr2+0** | **ptr2+1** | **ptr2+2** | **ptr2+3** | **ptr2+4** | | Address | 2000 | 2008 | 2016 | 2024 | 2032 | | Value | 0 | 0 | 0 | 0 | 0 | | 1. **Draw the recursive stack of the following function, if we call sum(3) with n = 3.**   int sum( int n)     {     if (n==0)  return 10;     else  return n + sum(n-1);  }  **returns 16** |
| 1. **Draw the recursive stack of the following function, if we call fibonacci(3) with n = 3.**   int fibonacci(int n)  {  if (n==0)  return 0;  else if (n==1)  return 1;  else    return fibonacci(n-2) + fibonacci(n-1);}  Solution:  Returns 2  Mentioned Below | 1. **Illustrate a memory allocation for the following structure object student1.**   struct day{    int date;char month[10];int year;};  struct student{   int id1, id2;   char a; float p;   struct day birthday;  }student1;  **Assume starting address as 1020**  **id1: 1020**  **id2: 1024**  **a: 1028**  **p:1032**  **date:1036**  **month:1040**  **year:1052** |
| 1. Considering the output, write down the missing part of the program. You must write only the missing part on the answer sheet with the most appropriate code. **CLO2** | |
| #include <stdio.h>  typedef struct{   int id; float price; char name[20];  }userTyped;  void main() {  userTyped inst1[]={{20, 5000.05,"Samsung"},                     {30, 3300.25, "Apple"},                     {40, 6020.05, "Acer"}};   userTyped \*ptr = inst1;      //Using variable      for(\_\_\_\_\_\_\_;\_\_\_\_\_\_\_;\_\_\_\_\_\_\_)          \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_      printf("------------\n");      //Using pointer      for(\_\_\_\_\_\_\_;\_\_\_\_\_\_\_;\_\_\_\_\_\_\_)          {  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_             \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_          }}  **SOlution:**  #include <stdio.h>  typedef struct{  int id; float price; char name[20];  }userTyped;  void main() {  userTyped inst1[]={{20, 5000.05,"Samsung"},  {30, 3300.25, "Apple"},  {40, 6020.05, "Acer"}};  userTyped \*ptr = inst1;  //Using variable  int i;  for(i = 2;i >=0;i--)  printf("%d %f %s\n", inst1[i].id, inst1[i].price, inst1[i].name);  printf("------------\n");  //Using pointer  for(i = 0;i <= 2;i++)  {  printf("%d %f %s\n", (ptr+i)->id, (ptr+i)->price, (ptr+i)->name);  }} | Output:  40, 6020.0, Acer  30, 3300.2, Apple  20, 5000.0, Samsung  ------------  20, 5000.0, Samsung  30, 3300.2, Apple  40, 6020.0, Acer |
| void main(){  char country[] = "Pakistan";  void \*ptr;  ptr = country;  while( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)    { \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     }}  **SOlution:**  #include<stdio.h>  int main(){  char country[] = "Pakistan";  void \*ptr;  ptr = country;  while( \*((char\*)(ptr)) != '\0')  {  printf("%c", \*((char\*)(ptr)));  ptr++;  }} | Output:  Pakistan |
| void main(){   char ch, \*str;   int cnt=0;   puts("enter any string: ");   while((ch=getche()) != 13){     if(cnt==0){   str = (char \*) malloc (sizeof(char));   str[cnt]=ch;}     else{  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  }  \_\_\_\_\_\_\_\_\_\_\_\_\_}   str[cnt]='\0';   printf("\n%s",str);  }  //Hint: You need to extend the dynamic array in this problem  **Solution:**  **#include<stdio.h>**  **#include<stdlib.h>**  **int main(){**  **char ch, \*str;**  **int cnt=0;**  **puts("enter any string: ");**  **while((ch=getche()) != 13){**  **if(cnt==0){**  **str = (char \*) malloc (sizeof(char));**  **str[cnt]=ch;}**  **else{**  **str = (char\*) realloc(str, (cnt+2)\*sizeof(char));**  **str[cnt] = ch;**    **}**  **cnt++;**  **}**  **str[cnt]='\0';**  **printf("\n%s",str);**  **}** | Output:  It will produce **“Pakistan Zindabad”** if input is **“Pakistan Zindabad”** |
| 1. Initialize and display the record structure:   struct employee{      int eid;  char ename[20];  };  struct date{     int joiningYear;};  struct record{    struct employee emp; struct date dt;  };  void main(){  struct record rcd[2]={  { \_\_\_\_\_\_\_\_\_\_\_\_\_\_,    \_\_\_\_\_\_\_\_\_\_\_\_\_\_  };        \_\_\_\_\_\_\_\_\_\_\_\_\_\_        \_\_\_\_\_\_\_\_\_\_\_\_\_\_  }}  **Solution:**  **#include<stdio.h>**  **#include<stdlib.h>**  **struct employee{**  **int eid; char ename[20];**  **};**  **struct date{**  **int joiningYear;};**  **struct record{**  **struct employee emp; struct date dt;**  **};**  **void main(){**  **struct record rcd[2]={**  **{{101,"Asad"}, 2010},**  **{{102,"Bilal"}, 2014}};**  **int i;**  **for(i = 0; i <2; i++){**  **printf("Employee ID: %d \nName: %s \nJoining Year: %d\n\n", rcd[i].emp.eid, rcd[i].emp.ename, rcd[i].dt.joiningYear);**  **}**  **}** | Output:  Employee ID:101  Name: Asad  Joining Year: 2010  Employee ID: 102  Name: Bilal  Joining Year: 2014 |
| void main() {  int arrAll[]={80, 82, 79, 71, 82, 65, 77};    for(\_\_\_\_\_\_\_; \_\_\_\_\_\_; \_\_\_\_\_\_\_\_)    {     for(\_\_\_\_\_\_\_; \_\_\_\_\_\_; \_\_\_\_\_\_\_\_)         \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_         \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_     }  }  Solution:  #include<stdio.h>  #include<stdlib.h>  int main() {  int i,j;  int arrAll[]={80, 82, 79, 71, 82, 65, 77};  for(i = 0; i < 7; i++)  {  for(j = 0; j <= i; j++)  printf("%c", arrAll[j]);    puts("");}  } | Output:  P  PR  PRO  PROG  PROGR  PROGRA  PROGRAM |
| void main(void){   char \*p[3] = {"Rashid", "Sajid", "Ali",};   char \* tmp; int i, j;    for( i = 0; i<3; i++)      for( \_\_\_\_\_; \_\_\_\_\_; \_\_\_\_\_)       {  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  { \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    }       }\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_        \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  }  Solution:  int main(void){  char \*p[3] = {"Rashid", "Sajid", "Ali",};  char tmp[20]; int i, j;  for(i=0; i<3; i++){  for(j=0; j<3-1-i; j++){  if(strcmp(p[j], p[j+1]) > 0){  //swap array[j] and array[j+1]  strcpy(tmp, p[j]);  strcpy(p[j], p[j+1]);  strcpy(p[j+1], tmp);  }  }  }  for( i = 0; i<3; i++)  puts(p[i]);    } | Output:  Ali  Rashid  Sajid |

**QUESTION:2 SOLUTION:**

#include<stdio.h>

int lighten(int image[3][3], int row, int col){

int rowCtr, colCtr;

for (rowCtr = 0; rowCtr < row; rowCtr++){

for(colCtr = 0; colCtr < col; colCtr++){

image[rowCtr][colCtr] \*= 1.10;

if(!(image[rowCtr][colCtr] >= 0 && image[rowCtr][colCtr] <= 255))

return 1;

}

}

return 0;

}

void display(int image[3][3], int row, int col){

puts("\nDisplaying the matrix after lightening");

int rowCtr, colCtr;

for (rowCtr = 0; rowCtr < row; rowCtr++){

for(colCtr = 0; colCtr < col; colCtr++){

printf("%d ", image[rowCtr][colCtr]);

}

puts("");

}

}

int main(){

int row, col, rowCtr, colCtr;

puts("Enter the number of rows and cols");

scanf("%d %d", &row, &col);

int image[row][col];

for (rowCtr = 0; rowCtr < row; rowCtr++){

for(colCtr = 0; colCtr < col; colCtr++){

do{

printf("Enter the row %d and col %d : \n", rowCtr, colCtr);

scanf("%d", &image[rowCtr][colCtr]);

}

while(!(image[rowCtr][colCtr] >= 0 && image[rowCtr][colCtr] <= 255));

}

}

if(lighten(image, row, col))

puts("Image is burnt out");

else

display(image, row, col);

}

QSolution.c

Open with ZIP Extractor

**QUESTION:3 SOLUTION:**

#include <stdio.h>

#include <String.h>

struct Customerlnfo{

char CustomerName[50];

char AddressName[50];

};

struct Car {

int Price;

int Model;

char Brand[50];

char ManufacturingDate[50];

char CountryOfOrigin[50];

struct Customerlnfo CI;

};

void printline() {

printf("\t-------------------------------------------\n");

}

long ServicesTax(int Price) {

return (Price \* 6) / 100;

}

long RetailProfit(int Price) {

return (Price \* 75) / 100;

}

long importDutyTax(int Price) {

return (Price \* 15) / 100;

}

long SalesTax(int Price) {

return 10 \* Price/ 100;

}

long CalulatePrice(int Price) {

long temp = SalesTax(Price) + ServicesTax(Price) + RetailProfit(Price) + importDutyTax(Price);

temp += Price;

return temp;

}

void PrintAllDetails(struct Car c) {

printline();

printf("\t\tBILLING DETAILS \n");

printline();

printf("\tImport Duty Cost: \tRs %ld \n",importDutyTax(c.Price));

printf("\tSales Tax Cost: \tRs %ld \n",SalesTax(c.Price));

printf("\tRetail Price: \t\tRs %ld \n",RetailProfit(c.Price));

printline();

printf("\tFinal Price: \t\tRs %ld \n",CalulatePrice(c.Price));

printf("\t\*\*\*\*\*\*\*\*THANKYOU FOR SHOPPING. \*\*\*\*\*\*\*\*\*\*\n\n\n");

}

void printBill(int model){

struct Car c;

FILE \*fptr = fopen("bill.txt","r");

if(fptr == NULL){

printf("Could not open file!");

}

else{

while(fread(&c, sizeof(struct Car), 1, fptr)){

if(c.Model == model){

printf("\t\tEnter CUSTOMER INFORMATION \n");

printline();

printf("\t\tCustomer Name: %s\n", c.CI.CustomerName);

printf("\t\tCustomer Address: %s\n", c.CI.AddressName);

printf("\t\tCard Brand: %s\n", c.Brand);

printf("\t\tCard Model: %d\n", c.Model);

printf("\t\tCar price: %d\n", c.Price);

printf("\t\tCountry of Origin: %s\n", c.CountryOfOrigin);

printf("\t\tCar Manufacturing Date: %s\n",c.ManufacturingDate);

PrintAllDetails(c);

}

}

fclose(fptr);

}

}

void SaveBillInfo(){

struct Car c;

puts("\t\tEnter Customer Name!");

printf("\t\t");

fflush(stdin);

scanf("%s", c.CI.AddressName);

puts("\t\tEnter Customer addres!");

printf("\t\t");

scanf("%s", c.CI.AddressName);

puts("\t\tEnter the price of Car!");

printf("\t\t");

scanf("%d", &c.Price);

puts("\t\tEnter the Model of Car!");

printf("\t\t");

scanf("%d", &c.Model);

puts("\t\tEnter the brand of car");

printf("\t\t");

fflush(stdin);

scanf("%s", c.Brand);

puts("\t\tEnter Manufacturing date of the car");

printf("\t\t");

scanf("%s", c.ManufacturingDate);

puts("\t\tEnter country of origin of the car!");

printf("\t\t");

scanf("%s", c.CountryOfOrigin);

FILE \*fptr = fopen("bill.txt","a");

if(fptr == NULL){

printf("Could not open file!");

}

else{

fwrite(&c, sizeof(struct Car), 1, fptr);

fclose(fptr);

}

}

void GetBillInfo(){

struct Car c;

FILE \*fptr = fopen("bill.txt","r");

if(fptr == NULL){

printf("Could not open file!");

}

else{

while(fread(&c, sizeof(struct Car), 1, fptr)){

printf("\t\tCustomer Name: %s\n", c.CI.CustomerName);

printf("\t\tCustomer Address: %s\n", c.CI.AddressName);

printf("\t\tCard Brand: %s\n", c.Brand);

printf("\t\tCard Model: %d\n", c.Model);

printf("\t\tCar price: %d\n", c.Price);

printf("\t\tCountry of Origin: %s\n", c.CountryOfOrigin);

printf("\t\tCar Manufacturing Date: %s\n",c.ManufacturingDate);

}

fclose(fptr);

}

}

int main(){

int choice =0;

outFile = fopen("Car.dat", "w+");

do{

printf("\n\n\tENTER CHOICE\n \t1.Save to Bill Details\n\t2.Get Bill Details\n\t3.Print All with taxes of perticular car\n");

printf("\tMake a choice: ");

fflush(stdin);

scanf("%d",&choice);

//printf("\n\n\n Checking %d \n\n\n",choice);

system("CLS");

if(choice==1){

//system("CLS");

SaveBillInfo();

} else if(choice==2){

GetBillInfo();

} else if (choice==3){

int model;

puts("\t\tEnter the Model No. for car");

printf("\t\t");

scanf("%d", &model);

printBill(model);

}

}while(choice!=0);

return 0;

}

**QUESTION:4SOLUTION:**

#include<stdio.h>

#include<stdlib.h>

struct group{

int groupID;

char groupName[20];

int tasks[5];

};

int sum(int \* arr){

int i; int sum = 0;

for(i = 0; i < 5; i++)

sum += arr[i];

return sum;

}

void diplayWinner(){

struct group gp;

int groupCtr = 1;

FILE \*fp = fopen("CompRecord1.txt","r");

if(fp == NULL){

puts("File Did not open!");

return;

}

else{

// reading the records

while(fread(&gp, sizeof(struct group), 1, fp)){

if(sum(gp.tasks) >= 3){

printf("\n\nWinner Group Details %d are\n\n", groupCtr);

printf("Group ID : %d\nGroup Name : %s\n", gp.groupID, gp.groupName);

puts("");

groupCtr++;

}

}

fclose(fp);

}

}

void search(){

int ID, counter, isFound = 0;

FILE \*fp = fopen("CompRecord1.txt","r");

if(fp == NULL){

puts("File Did not open!");

return;

}

puts("Enter the Group ID you want to search!");

scanf("%d", &ID);

else{

struct group gp;

// reading the records

while(fread(&gp, sizeof(struct group), 1, fp)){

if(gp.groupID == ID){

puts("\n\nGroup Details are\n");

printf("Group ID : %d\nGroup Name : %s\n", gp.groupID, gp.groupName);

puts("Task status is ");

for(counter = 0; counter < 5; counter++){

printf("Task %d : %d \n",counter+1, gp.tasks[counter]);

}

isFound = 1;

}

}

if(isFound == 0)

puts("ID not Found");

fclose(fp);

}

}

void input(){

struct group gp;

int ctr;

puts("Enter Group ID");

scanf("%d", &gp.groupID);

fflush(stdin);

puts("Enter Group Name");

scanf("%s", gp.groupName);

puts("Enter the results of tasks 1 for pass o for fail");

for(ctr = 0; ctr <5; ctr++){

printf("Enter value of Task %d :", ctr+1 );

scanf("%d", &gp.tasks[ctr]);

if(gp.tasks[ctr] != 1 && gp.tasks[ctr] != 0){

puts("Re-Enter the value either 0 or 1");

ctr--;

}

}

FILE \*fp = fopen("CompRecord1.txt","a");

if(fp == NULL){

puts("File Did not open!");

return;

}

else{

fwrite(&gp, sizeof(gp), 1, fp);

fclose(fp);

}

}

int main(){

// remove("CompRecord1.txt");

char choice = 'Y';

int op;

while(choice == 'Y' || choice == 'y'){

puts("Enter 1 to input the record\nEnter 2 to Display Winner\nEnter 3 to search");

scanf("%d", &op);

switch(op){

case 1:

input();

break;

case 2:

diplayWinner();

break;

case 3:

search();

break;

default:

puts("Invalid Value");

}

puts("Enter Y to Continue!! Press any key to exit");

fflush(stdin);

scanf("%c", &choice);

}

}