**The shop CSV should hold the initial cash value for the shop.**

|  |
| --- |
| No Code |
| The input CSV file is created in MS Excel, with the name “stock.csv”. Its first row contains the initial cash value for the shop.  It is not necessary to create the CSV file in MS Excel, it can also be created using the Notepad.  <https://www.computerhope.com/issues/ch001356.htm> |

**Read in customer orders from a CSV file.**

|  |
| --- |
| struct Customer readCustomer(char file\_name[MAX\_FILE\_NAME\_LENGTH],struct Shop \*s)  {      FILE \* fp;      char \* line = NULL;      size\_t len = 0;      size\_t read;      fp = fopen(file\_name, "r");      if (fp == NULL)          exit(EXIT\_FAILURE);        read = getline(&line, &len, fp);      char \*n = strtok(line, ",");      char \*b = strtok(NULL, ",");      char \*customer\_name = malloc(sizeof(char) \* 50);      strcpy(customer\_name, n);      double customer\_budget = atof(b);        struct Customer c = {.name = customer\_name, .budget = customer\_budget};      while ((read = getline(&line, &len, fp)) != -1)      {          char \*n = strtok(line, ",");          char \*q = strtok(NULL, ",");          int quantity = atoi(q);          char \*name = malloc(sizeof(char) \* 50);          strcpy(name, n);          double price = findProductPrice(name,s);          struct Product product = { name, price };          struct ProductStock product\_stock = { product, quantity };          c.shoppingList[c.index++] = product\_stock;      }        return c;  } |
| <https://www.programiz.com/c-programming/c-file-input-output>  <https://www.programiz.com/c-programming/examples/read-file>  <https://www.tutorialspoint.com/cprogramming/c_strings.htm> |

**That file should include all the products they wish to buy and in what quantity.**

|  |
| --- |
| No Code |
| There are 3 different customer files in the folder, for 3 different customers. The file names are as follows:  customer.csv  customer\_not\_enough\_money.csv  customer\_too\_much\_bread.csv  Each file has same format. The first row contains the customer’s name followed by his budget. The next rows contains the product that the customer wants to buy, followed by the product’s quantity in the same row. |

**It should also include their name and their budget.**

|  |
| --- |
| No Code |
| The first row of the customers’ CSV files contains the customer’s name followed by his budget. |

**The shop must be able to process the orders of the customer.**

|  |
| --- |
| void printCustomer(struct Customer c, struct Shop \*s)  {      printf("CUSTOMER NAME: %s \nCUSTOMER BUDGET: %.2f\n\n", c.name, c.budget);      double total = 0.0;      double cost = 0.0;      struct temp\_out\_of\_stock out\_of\_stock={.index=0};      struct temp\_active\_products active\_products\_list={.index=0};      for(unsigned int i = 0; i < c.index; i++)      {          //check whether the product exisits in stock or not          unsigned int check\_val = findProduct(c.shoppingList[i].product.name,s);          if (check\_val == 0)          {              //printf("\n");              printf("Error no %s in Shop stock. List item ignored\n",c.shoppingList[i].product.name);          }          else          {              printProduct(c.shoppingList[i].product);              printf("\n%s ORDERS %d OF ABOVE PRODUCT\n", c.name, c.shoppingList[i].quantity);              if (findProductQuantity(c.shoppingList[i].product.name,s) < c.shoppingList[i].quantity)              {                  struct temp\_data data = {.product\_name=c.shoppingList[i].product.name,                                          .available\_qty=findProductQuantity(c.shoppingList[i].product.name,s),                                          .required\_qty=c.shoppingList[i].quantity};                  out\_of\_stock.products[out\_of\_stock.index] = data;                  out\_of\_stock.index = (out\_of\_stock.index)+1;              }              cost = c.shoppingList[i].quantity \* c.shoppingList[i].product.price;              struct temp\_buying\_data buying\_data = {.product\_name=c.shoppingList[i].product.name,                                                      .qty=c.shoppingList[i].quantity,                                                      .sub\_total=cost};              active\_products\_list.products[active\_products\_list.index] = buying\_data;              active\_products\_list.index = (active\_products\_list.index)+1;              total+=cost;              printf("The cost to %s will be $%.2f\n\n", c.name, cost);          }      }      printf("--------------------------------------------------\n");      printf("The total cost to %s will be $%.2f\n", c.name, total);      if (out\_of\_stock.index == 0)      {          if (c.budget >= total)          {              printf("--------------------------------------------------\n");              printf("SUCCESS !\n");              printf("%s's budget is %.2f\n",c.name,c.budget);              printf("%s has enough money\n",c.name);              printf("total of %.2f will be deducted from %s's budget\n",total,c.name);              //updating the stock              for(unsigned int k = 0; k < s->index; k++)              {                  for(unsigned int m = 0; m < active\_products\_list.index; m++)                      if (strcmp(s->stock[k].product.name, active\_products\_list.products[m].product\_name)==0)                          s->stock[k].quantity = (s->stock[k].quantity - active\_products\_list.products[m].qty);              }                c.budget = c.budget - total;              s->cash = s->cash + total;              printf("%s's budget is %.2f\n",c.name,c.budget);              printf("total of %.2f added to shop\n",total);              printf("Shop has %.2f\n",s->cash);          }          else          {              printf("--------------------------------------------------\n");              printf("FAIL !\n");              printf("%s's budget is %.2f\n",c.name,c.budget);              printf("The total cost of all items is ...! %.2f\n",total);              printf("%s does not have enough money\n",c.name);              printf("total of 0 will be deducted from %s's budget\n",c.name);              printf("John's budget is %.2f\n",c.budget);              printf("total of 0 added to shop\n");              printf("Shop has %.2f in cash\n",s->cash);          }      }      else      {          for(unsigned int j = 0; j < out\_of\_stock.index; j++)          {              printf("Not enough %s in stock\n",out\_of\_stock.products[j].product\_name);              printf("%d %s in stock\n",out\_of\_stock.products[j].available\_qty,out\_of\_stock.products[j].product\_name);              printf("%s want %d %s\n",c.name,out\_of\_stock.products[j].required\_qty,out\_of\_stock.products[j].product\_name);              printf("Please revise order and upload again!\n");          }      }  } |
| This function reads the customer’s CSV file and process each of the ordered product in following steps:   1. First of all, it checks if the ordered product exists in shop’s stock or not 2. If the product exists, it checks if there is sufficient quantity of that product available in stock in order to meet consumer’s required quantity. 3. It then fetches the product’s price from the shop’s stock 4. It multiplies the product price with the ordered quantity to get the total cost of that product 5. After going through each of the products in consumer’s order, it then checks if the grand total of the consumer’s order is within consumer’s budget 6. If the order is within budget, the order is processed. The respective ordered products’ quantities are removed from the stock and the order’s total value is added in the shop’s total cash value. |

**Update the cash in the shop based on money received.**

|  |
| --- |
| if (c.budget >= total)  {      printf("--------------------------------------------------\n");      printf("SUCCESS !\n");      printf("%s's budget is %.2f\n",c.name,c.budget);      printf("%s has enough money\n",c.name);      printf("total of %.2f will be deducted from %s's budget\n",total,c.name);      //updating the stock      for(unsigned int k = 0; k < s->index; k++)      {          for(unsigned int m = 0; m < active\_products\_list.index; m++)              if (strcmp(s->stock[k].product.name, active\_products\_list.products[m].product\_name)==0)                  s->stock[k].quantity = (s->stock[k].quantity - active\_products\_list.products[m].qty);      }        c.budget = c.budget - total;      s->cash = s->cash + total;      printf("%s's budget is %.2f\n",c.name,c.budget);      printf("total of %.2f added to shop\n",total);      printf("Shop has %.2f\n",s->cash);  }  else  {      printf("--------------------------------------------------\n");      printf("FAIL !\n");      printf("%s's budget is %.2f\n",c.name,c.budget);      printf("The total cost of all items is ...! %.2f\n",total);      printf("%s does not have enough money\n",c.name);      printf("total of 0 will be deducted from %s's budget\n",c.name);      printf("John's budget is %.2f\n",c.budget);      printf("total of 0 added to shop\n");      printf("Shop has %.2f in cash\n",s->cash);  } |
| If the consumer order’s total value is within consumer’s budget, then the order is processed and the total amount of consumer’s order is added to shop’s cash, and a success message is shown on screen. Otherwise, a fail message is displayed and no amount is added to the shop’s cash value. |

**It is important that the state of the shop be consistent.**

|  |
| --- |
| //updating the stock  for(unsigned int k = 0; k < s->index; k++)  {      for(unsigned int m = 0; m < active\_products\_list.index; m++)          if (strcmp(s->stock[k].product.name, active\_products\_list.products[m].product\_name)==0)              s->stock[k].quantity = (s->stock[k].quantity - active\_products\_list.products[m].qty);  }    c.budget = c.budget - total;  s->cash = s->cash + total;  printf("%s's budget is %.2f\n",c.name,c.budget);  printf("total of %.2f added to shop\n",total);  printf("Shop has %.2f\n",s->cash); |
| When the consumer’s order is processed, the shop’s stock of each product is updated accordingly to ensure that the state of the shop remains consistent after each order. |

**You should create customer test files (CSVs) which cannot be completed by the shop e.g. customer wants 400 loaves of bread but the shop only has 20, or the customer wants 2 cans of coke but can only afford 1.**

|  |
| --- |
| No Code |
| 2 separate CSV files are created with the following names:  customer\_not\_enough\_money.csv  customer\_too\_much\_bread.csv |

**– Know whether or not the shop can fill an order.**

|  |
| --- |
| //check whether the product exisits in stock or not  unsigned int check\_val = findProduct(c.shoppingList[i].product.name,s);  if (check\_val == 0)  { printf("Error no %s in Shop stock. List item ignored\n",c.shoppingList[i].product.name);  }  for(unsigned int j = 0; j < out\_of\_stock.index; j++)  {  printf("Not enough %s in stock\n",out\_of\_stock.products[j].product\_name);  printf("%d %s in stock\n",out\_of\_stock.products[j].available\_qty,out\_of\_stock.products[j].product\_name);  printf("%s want %d %s\n",c.name,out\_of\_stock.products[j].required\_qty,out\_of\_stock.products[j].product\_name);  printf("Please revise order and upload again!\n");  } |
| The first code checks if the ordered product exists in the stock or not  The second code checks if the ordered product’s quantity is available in stock or not |

**\* Thrown an appropriate error.**

|  |
| --- |
| printf("NameError: Please Use Numbers Only\n");  printf("Error no %s in Shop stock. List item ignored\n",c.shoppingList[i].product.name);  exit(EXIT\_FAILURE); |
| The first error message is thrown in “main” function, when the user enters an incorrect choice  The second error message is thrown in the “printCustomer” function when entered product doesn’t exist in stock  The third error is thrown at several positions in the code, whenever the code fails to open an input csv file. This is in fact a program exit code. More details on: <https://en.cppreference.com/w/c/program/EXIT_status> |

**Operate in a live mode, where the user can enter a product by name, specify a quantity, and pay for it. The user should be able to buy many products in this way.**

|  |
| --- |
| case 1:      printf("---------------------------------------------------\n");      printf("Live mode\n");      printf("Type your budget : ");      scanf("%lf",&custom\_budget);      while (custom\_active == 1)      {          char custom\_product[MAX\_FILE\_NAME\_LENGTH]="";          printf("Enter a product name you would like? Eg: Coke Can, Bread, Spaghetti, Tomato Sauce, Big Bags : ");          //scanf("%s",custom\_product);          //fgets(custom\_product,50,stdin);          scanf(" %[^\n]s",custom\_product);          if (findProduct(custom\_product,&shop)!=1)          {              printf("This product : %s, is not in stock\n",custom\_product);          }          else          {              printf("Entered Product is %s \n",custom\_product);              int custom\_count=0;              double custom\_sub\_total = 0.0;              int available\_qty = findProductQuantity(custom\_product,&shop);              printf("How many do you want? : ");              scanf("%d",&custom\_count);              if (available\_qty >= custom\_count)              {                  custom\_sub\_total = (findProductPrice(custom\_product,&shop) \* custom\_count);                  printf("The cost to you will be %.2f\n",custom\_sub\_total);                  struct temp\_buying\_data buying\_data = {.product\_name=custom\_product,                                                  .qty=custom\_count,                                                  .sub\_total=custom\_sub\_total};                  active\_products\_list.products[active\_products\_list.index] = buying\_data;                  active\_products\_list.index = (active\_products\_list.index)+1;              }              else              {                  printf("\n");                  printf("FAIL ! \n");                  printf("Not enough %s in stock\n",custom\_product);                  printf("%d %s in stock\n",available\_qty,custom\_product);                  printf("You want %d of %s\n",custom\_count, custom\_product);                  printf("Please revise order and try again!\n");              }          }          char close\_tag[4] = "Y";          printf("Would you like another product? Y/N ");          scanf("%s",close\_tag);          if ((strcmp(close\_tag,"N")==0) || (strcmp(close\_tag,"n")==0))          {              custom\_active = 0;          }            //if ((close\_tag == 'N') || (close\_tag == 'n'))          //  custom\_active = 0;      }      printf("Summary\n");      if (active\_products\_list.index > 0)      {          double total = 0;          for(unsigned int m = 0; m < active\_products\_list.index; m++)              total = total + active\_products\_list.products[m].sub\_total;          if (total > custom\_budget)          {              printf("FAIL! \n");              printf("Your budget is ...\n");              printf("The total cost of all items is ...! %.2f\n",total);              printf("You do not have enough money\n");              printf("Total of 0 will be deducted from your budget\n");              printf("Your budget is %.2f\n",custom\_budget);              printf("Total of 0 added to shop\n");              printf("Shop has %.2f in cash\n",shop.cash);          }          else          {              printf("SUCCESS! \n");              printf("Your budget is %.2f\n",custom\_budget);              printf("You have enough money\n");              printf("Total of %.2f will be deducted from your budget\n",total);              printf("Total of %.2f will be added to shop\n",total);              //updating the stock              for(unsigned int k = 0; k < shop.index; k++)              {                  for(unsigned int m = 0; m < active\_products\_list.index; m++)                      if (strcmp(shop.stock[k].product.name, active\_products\_list.products[m].product\_name)==0)                          shop.stock[k].quantity = (shop.stock[k].quantity - active\_products\_list.products[m].qty);              }              double shop\_total = 0.0;              shop\_total = shop.cash + total;              shop.cash = shop\_total;              custom\_budget = custom\_budget - total;              //printf(shop.stock[0].product.name, shop.stock[0].quantity);              printf("Shop has %.2f in cash\n",shop\_total);          }      }      else      {          printf("Something went wrong\n");      }      printf("---------------------------------------------------\n");      break; |
| Taking Input and Printing Output:  <https://www.programiz.com/c-programming/c-input-output>  Structures:  [https://www.studytonight.com/c/structures-in-c.php#](https://www.studytonight.com/c/structures-in-c.php)  Pointers to structures:  <https://www.programiz.com/c-programming/c-structures-pointers> |