اونيۏرسيتي مليسيا ڤهڠ

UNIVERSITI MALAYSIA PAHANG

PROJECT BCI1093 DATA STRUCTURE & ALGORITHMS

TITLE: SUPERMARKET BILLING SYSTEM (GROUP 6)

LECTURER'S NAME: DR. JUNAIDA BINTI SULAIMAN

Matric ID	Name	Section
SD21063	TEAN JIN HE	02G
SD21038	LIM JOEY	02G
SD21039	CHONG JIA XIN	02G
SD21043	LING HWEI ERN, DEBORAH	02G
SD21044	DAVID LAU KING LUEN	02G

Table of Content

CASE STUDY 1: SUPERMARKET BILLING SYSTEM (NEVON PROJECTS)

INTRODUCTION

Supermarket building system has a traditional supermarket building system with some added functionality. This system is used for data processing and bill generation for supermarket customers. When a product is scanned, the price is added based on the product quantity. The system can also calculate discounts for various products when billing. There are two papers that will log into the system where one will be the admin and the one will be the cashier. The page will pop up for admin to enter admin ID and password. Then, the admin will add in the cashier details and also the password. Next, is to enter the product details. When the information is entered, the details of the cashier login time, logout time, date and amount will be displayed. The admin can also view transactions, inventory, update stock and check the sales per product. All these details can be exported to an Excel file. For the cashier login, they will need to enter the bill number, product ID, product name, product price, product quantity and product discount for the system to calculate the total amount.

PROBLEM

The problem with this system is that the admin needs to find the details one by one.

SOLUTION

The details are all listed in one list.

CASE STUDY 2: LOTUS SUPERMARKET BILLING SYSTEM

INTRODUCTION

For Lotus Supermarket, they have an online store where their customers can buy their products online and delivered to their house without the need to leave their house to get groceries. Although a lot of people prefer to buy things online, there are also people who would like to get their items in the real shop. This is because it enables them to choose fresh products on their own and not products that are not fresh sent to them.

Through the online system, customers are able to choose whether they want the product to be sent to them or they want to pick up their product on their own. For the delivery system, some areas are not in the delivery coverage area so the customers are unable to get their product from their home. Therefore, they still need to buy the products at the retail store. At the retail store, the products are scanned using the barcode on the product when the customer wants to make payment at the counter. Before that, the cashier will ask them whether they have a member card or not. If yes, then the cashier will proceed to scan the card to get the customer information so that they're able to earn points from their purchase.

PROBLEM

The issue that they had is that the delivery is unable to be sent to every place as some delivery addresses are not supported.

SOLUTION

They need a wider delivery coverage area.

CASE STUDY 3: GROCERY STORE BILLING SOFTWARE

INTRODUCTION

Mr.Sharma runs a grocery business. He constantly faces difficulties while handling extremely varied products with different lifespans which complicates his ordering and inventory management. His store always has long queues as billing of many items needs to be done every minute. Most of his stocks are dumped as he is not aware of the amount of stocks that he has in his store. He couldn't keep up with the discounts and schemes for business. His new employee does not know where the items are placed on the rack which frustrates his customers who are in the queue to wait for their order. Mr.Sharma also finds it hard to remember when he has to avail offers from his distributors before a certain time limit. When he remembered, he had already lost the business. He needs to take care of a lot of things such as reorder and purchase order, expiry stock management, accounts, files, GST returns and many other tasks. Mr. Sharma realized that he needs to implement new technology for his business. He found MargERP for his business and it has helped him a lot.

PROBLEM

Mr. Sharma faces problems in his billing system as his traditional system needs to key in the item one by one.

He also had to do all the paperwork by himself using a filing system and not using technology.

SOLUTION

He reached out to MargERP for his business so that they can implement the new billing system into his grocery business.

1.0 Project Case Study

Nowadays, supermarkets are no longer uncommon among us. This is because we need to buy daily essentials from supermarkets. From youngsters to elderly, everyone had gone to the supermarket. Supermarkets have also implemented technology in their billing system. The billing system that had been implemented in supermarkets made the work to be done faster and easier. The modern billing system in the supermarket even involves self-payment kiosks nowadays without the need to wait in line.

From the three case studies that we have found, we are able to create an ideal and user friendly system for the cashier to use. Our billing system is able to list out all the customer details and product details in one list so that the admin or cashier would not need to look for the details one by one other than that we also implemented the sorting for product quantity, product price list and also according to the product number. This could help the admin to write their report easily as the product will be arranged by the system automatically.

Other than that, our system is also able to be used to update customer details and product details when it is needed as well as deletes user and item. We can also update new customers and new products with no trouble as the supermarket will always have increments in customer and product.

2.0 CODING

```
/*DSA Group 6 Assignment*/
/*TEAN JIN HE*/
/*DAVID LAU KING LUEN*/
/*LIM JOEY*/
/*CHONG JIA XIN*/
/*LING HWEI ERN, DEBORAH*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include <conio.h>
#include <ctype.h>

#define TRUE 1
#define FALSE 0
#define MAX 50
#define NUM 15

struct date
{
   int month,day,year;
};

struct member
{
   int membercard;
};

struct customer
{
   char name[MAX], phonenum[NUM], address[MAX];
   struct member mm;
   struct customer *customerlink;
};

struct sorting
{
   int sproduct_number;
   char sproduct_name[MAX];
   float
sproduct_price,sproduct_quantity,sproduct_tax,sproduct_gross,sproduct_discount,sproduct_net_amou
nt;
};

struct product
{
```

```
    int product_number;
    char product_name[MAX];
    float
product_price,product_quantity,product_tax,product_gross,product_discount,product_net_amount;
    struct date d;
    struct product *link;
};

struct customer *customerheadptr, *addcustomernewptr, *customercurrentptr, *customerprevptr,
*customertempptr;
struct product *productheadptr, *addproductnewptr, *productcurrentptr, *productprevptr,
*deleteproductptr;

void gotoxy(int x, int y)//goes to x,y console
{
    COORD coord;
    coord.X = x;
    coord.Y = y;
    SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

void sleep(long int interval)
{
    for(int i=0;i<interval*1000;i++);
}

void customerdetails()
{
    system("cls");
    int x;
    printf("\tCUSTOMER DETAILS \n");
    printf("\n-------------------------------------------------\n");
    printf("[1] - ADD CUSTOMER \n");
    printf("[2] - UPDATE CUSTOMER\n");
    printf("[3] - DELETE CUSTOMER\n");
    printf("[4] - CUSTOMER LIST\n");
    printf("[5] - RETURN MAIN MENU\n");
    printf("\n-------------------------------------------------\n");
    printf("Which option you want to choose? ");
    scanf("%d",&x);
    switch(x)
    {
        case 1 :addcustomer();break;
        case 2 :updatecustomer();break;
        case 3 :deletecustomer();break;
        case 4 :customerlist();break;
        case 5 :return main();break;
        default:printf("\nWrong Option....Please Retry!");getch();return customerdetails();
```

```
    }
}

void addcustomer()
{
    system("cls");
    addcustomernewptr=(struct customer *)malloc(sizeof (struct customer));
    char ch;
    printf("Enter Name: ");
    fflush(stdin);
    gets(addcustomernewptr -> name);
    printf("Enter Phone Number: ");
    fflush(stdin);
    gets(addcustomernewptr -> phonenum);
    printf("Enter Address: ");
    fflush(stdin);
    gets(addcustomernewptr -> address);
    printf("Did you have membercard (Y/N)? ");
    scanf(" %c", &ch);
    do
    {
        if (ch=='Y'|| ch=='y')
        {
            printf("Enter your membercard number: ");
            scanf("%d", &addcustomernewptr -> mm.membercard);
        }
        else if (ch=='N'|| ch=='n')
        {
            addcustomernewptr -> mm.membercard = NULL;
        }
        else
        {
            printf("INVALID CHOICE, PLEASE TRY AGAIN...");
        }
    }
    while (ch!='Y' && ch!='y' && ch!='N' && ch!='n');
    {
        if (customerheadptr==NULL)
        {
            customerheadptr = addcustomernewptr;
            addcustomernewptr->customerlink = NULL;
        }
        else
        {
            customercurrentptr = customerheadptr;
            while (customercurrentptr->customerlink != NULL )
            {
                customercurrentptr = customercurrentptr->customerlink;
```

```
            }
            customercurrentptr->customerlink = addcustomernewptr;
            addcustomernewptr->customerlink = NULL;
        }
    }
    printf("\nCUSTOMER HAS BEEN ADDED!!");
    getch();
    return customerdetails();
}

void updatecustomer()
{
    system("cls");
    char customername[MAX],ch;
    customercurrentptr = customerheadptr;
    if (customercurrentptr == NULL)
    {
        printf("\nEmpty List");
        getch();
        return customerdetails();
    }
    else
    {
        printf("Enter name that you want to update: ");
        fflush(stdin);
        gets(customername);
        while(customercurrentptr!=NULL && strcmp(customercurrentptr->name,customername)<0)
        {
            customerprevptr = customercurrentptr;
            customercurrentptr = customercurrentptr->customerlink;
        }
        if(customercurrentptr != NULL)
        {
            printf("Name : %s", customercurrentptr->name);
            printf("\n-----------------------------------------\n");
            printf("\n\tUPDATE\n");
            printf("Enter customer new phone number: ");
            fflush(stdin);
            gets(customercurrentptr->phonenum);
            printf("Enter customer new address: ");
            fflush(stdin);
            gets(customercurrentptr->address);
            printf("Did you have membercard (Y/N)? ");
            scanf(" %c", &ch);
            do
            {
                if (ch=='Y'|| ch=='y')
                {
```

```
                printf("Enter your new membercard number: ");
                scanf("%d",&customercurrentptr->mm.membercard);
            }
            else if (ch=='N'|| ch=='n')
            {
                customercurrentptr->mm.membercard = NULL;
            }
            else
            {
                printf("INVALID CHOICE, PLEASE TRY AGAIN...");
            }
        }
        while (ch!='Y' && ch!='y' && ch!='N' && ch!='n');
    }
    else
    {
        printf("customer not found");
        getch();
        return customerdetails();
    }
  }
}

void deletecustomer()
{
  system("cls");
  char name[MAX];
  if (customerheadptr==NULL)
        {
          system("cls");
                printf("\nEmpty list");
                getch();
                return customerdetails();
        }
        else
  {
    printf("\n\nEnter name to be deleted: ");//Prompt customer to enter name to be deleted
    fflush(stdin);
    gets(name);

    struct customer *current = customerheadptr;
    struct customer *previous = NULL;

    while(current!=NULL)
    {
      if (strcmp(name,current->name)==0) //found the location
      {
        if (previous == NULL)
```

```
            {
               customerheadptr = current->customerlink;
            }
            else
            {
               previous->customerlink = current->customerlink;
            }
          printf("Name : %s",current->name);
          free(current);
          printf("\n\nSUCCESSFULLY DELETED!!");
          getch();
          return customerdetails();
          }
        previous = current;
        current = current->customerlink;
        }
    printf("\nName not found.");
    getch();
    return customerdetails();
    }
}

void customerlist()
{
    int i=0;
    if (customerheadptr==NULL)
            {
               system("cls");
                     printf("\nEmpty list");
                     getch();
                     return customerdetails();
            }
           customercurrentptr = customerheadptr;
    system("cls");

    gotoxy(30,3);
    printf(" CUSTOMER DETAILS ");
    gotoxy(3,5);
    printf("NAME");
    gotoxy(23,5);
    printf("ADDRESS");
    gotoxy(75,5);
    printf("PHONE NUMBER");
    gotoxy(95,5);
    printf("MEMBERCARD");
    int k=7;
    do
    {
```

```
        gotoxy(3,k);
        printf("%s", customercurrentptr->name);
        gotoxy(23,k);
        printf("%s", customercurrentptr->address);
        gotoxy(75,k);
        printf("%s",customercurrentptr->phonenum);
        gotoxy(95,k);
        printf("%d",customercurrentptr->mm.membercard);
        k=k+1;
                    customercurrentptr = customercurrentptr->customerlink;
            }
            while(customercurrentptr != NULL);
            getch();
}

void productdetails()
{
   system("cls");
   struct sorting sort[200];

   int y;
   printf("\tPRODUCT DETAILS \n");
   printf("\n-------------------------------------------------\n");
   printf("[1] - ADD PRODUCT \n");
   printf("[2] - UPDATE PRODUCT\n");
   printf("[3] - DELETE PRODUCT\n");
   printf("[4] - PRODUCT LIST\n");
   printf("[5] - RETURN MAIN MENU\n");
   printf("\n-------------------------------------------------\n");
   printf("Which option you want to choose? ");
   scanf("%d",&y);
   switch(y)
   {
      case 1 :addproduct();break;
      case 2 :updateproduct(sort);break;
      case 3 :deleteproduct();break;
      case 4 :productlist();break;
      case 5 :return main();break;
      default:printf("\nWrong Option....Please Retry!");getch();return productdetails();
   }
}


void addproduct()
{
   system("cls");

   addproductnewptr=(struct product *)malloc(sizeof (struct product));
```

```
   printf("\nEnter product number : ");
         scanf("%d",&addproductnewptr->product_number);
         printf("\nEnter product name : ");
         fflush(stdin);
   gets(addproductnewptr->product_name);
         printf("\nEnter date(dd-mm-yy): ");

scanf("%d-%d-%d",&addproductnewptr->d.day,&addproductnewptr->d.month,&addproductnewptr->
d.year);
   printf("\nEnter product price: ");
         scanf("%f",&addproductnewptr->product_price);
         printf("\nEnter product quantity: ");
         scanf("%f",&addproductnewptr->product_quantity);
         printf("\nEnter tax percent: ");
         scanf("%f",&addproductnewptr->product_tax);
         printf("\nEnter product discount: ");
         scanf("%f",&addproductnewptr->product_discount);

   addproductnewptr->product_gross =
addproductnewptr->product_price+(addproductnewptr->product_price*(addproductnewptr->product_
tax/100));
   addproductnewptr->product_net_amount =
addproductnewptr->product_quantity*(addproductnewptr->product_gross-(addproductnewptr->produ
ct_gross*(addproductnewptr->product_discount/100)));
   addproductnewptr->link = productheadptr;//stack statement of push
   productheadptr = addproductnewptr;
   printf("\n\nPRODUCT HAD BEEN ADDED!!");
   getch();
}

void deleteproduct()
{
   system("cls");
   if (productheadptr == NULL)
   {
      printf("\nEmpty list");
      getch();
      return productdetails();
   }
   else
   {
      int productNumber;
      printf("Enter the product number of the product to delete: ");
      scanf("%d", &productNumber);

      struct product *productcurrentptr = productheadptr;
      struct product *productprevptr = NULL;
```

```
        while (productcurrentptr != NULL)
            {
            if (productcurrentptr->product_number == productNumber)
                {
                if (productprevptr == NULL)
                {
                    productheadptr = productcurrentptr->link;
                }
                else {
                    productprevptr->link = productcurrentptr->link;
                }
                printf("\n Product Number: %d\n Product Name: %s\n Product Price: RM%.2f\n Product
Quantity: %.f\n Product Tax: %.f\n Product Gross: %.f\n Product Discount: %.f\n Product Net
Amount: RM%.2f",
productcurrentptr->product_number,productcurrentptr->product_name,productcurrentptr->product_pr
ice,

productcurrentptr->product_quantity,productcurrentptr->product_tax,productcurrentptr->product_gros
s,productcurrentptr->product_discount,productcurrentptr->product_net_amount);
                free(productcurrentptr);
                printf("\n\nSUCCESSFULLY DELETED!!");
                getch();
                return productdetails();
            }
            productprevptr = productcurrentptr;
            productcurrentptr = productcurrentptr->link;
            }
        printf("\nproduct not found.");
        getch();
        return productdetails();
}
}


void productlist()
{
    struct sorting sort[200];
    system("cls");
    int s,i=0;
    printf("\tPRODUCT LIST \n");
    printf("\n-------------------------------------------------\n");
    printf("[1] - SHOW PRODUCT \n");
    printf("[2] - ARRANGE BY PRICE\n");
    printf("[3] - ARRANGE BY QUANTITY\n");
    printf("[4] - ARRANGE BY PRODUCT NUMBER\n");
    printf("[5] - RETURN MAIN MENU\n");
    printf("\n-------------------------------------------------\n");
```

```
   printf("Which option you want to choose? ");
   scanf("%d",&s);
   switch(s)
   {
      case 1 :showproduct();break;
      case 2 :pricelist(sort,i);break;
      case 3 :quantitylist(sort,i);break;
      case 4 :numberlist(sort,i);break;
      case 5 :return productdetails();break;
      default:printf("\nWrong Option....Please Retry!");getch();return productlist();
   }
}

void showproduct()
{
   if (productheadptr==NULL)
         {
            system("cls");
                  printf("\nEmpty list");
                  getch();
                  return;
         }
         productcurrentptr = productheadptr;
   system("cls");

   gotoxy(30,3);
   printf(" PRODUCT DETAILS ");
   gotoxy(3,5);
   printf("NUMBER");
   gotoxy(13,5);
   printf("NAME");
   gotoxy(23,5);
   printf("PRICE(RM)");
   gotoxy(33,5);
   printf("QUANTITY");
   gotoxy(44,5);
   printf("TAX");
   gotoxy(52,5);
   printf("DISCOUNT(%)");
   gotoxy(64,5);
   printf("NET AMOUNT(RM)");
   gotoxy(82,5);
   printf("DATE");
   int k=7;
   do
   {
      gotoxy(3,k);
      printf("%d", productcurrentptr->product_number);
```

```
        gotoxy(13,k);
        printf("%s", productcurrentptr->product_name);
        gotoxy(23,k);
        printf("RM%.2f",productcurrentptr->product_price);
        gotoxy(33,k);
        printf("%.f",productcurrentptr->product_quantity);
        gotoxy(44,k);
        printf("%.f",productcurrentptr->product_tax);
        gotoxy(52,k);
        printf("%.f",productcurrentptr->product_discount);
        gotoxy(64,k);
        printf("%.2f",productcurrentptr->product_net_amount);
        gotoxy(82,k);
        printf("%d-%d-%d", productcurrentptr->d.day, productcurrentptr->d.month,
productcurrentptr->d.year);
        k=k+1;
                        productcurrentptr = productcurrentptr->link;
            }
            while(productcurrentptr != NULL);
            getch();
}

void pricelist(struct sorting sort[200],int i)
{
   int smallest;
   int j,k;
   struct sorting temp;
   if (productheadptr==NULL)
            {
               system("cls");
                        printf("\nEmpty list");
                        getch();
                        return productdetails();
            }
   else
   {
      system("cls");
      productcurrentptr=productheadptr;
      do
      {
         sort[i].sproduct_number = productcurrentptr->product_number;
         strcpy(sort[i].sproduct_name,productcurrentptr->product_name);
         sort[i].sproduct_price = productcurrentptr->product_price;
         sort[i].sproduct_quantity = productcurrentptr->product_quantity;
         sort[i].sproduct_tax = productcurrentptr->product_tax;
         sort[i].sproduct_discount = productcurrentptr->product_discount;
         sort[i].sproduct_net_amount = productcurrentptr->product_net_amount;
         productcurrentptr = productcurrentptr -> link;
```

```
            i++;
        }
        while(productcurrentptr != NULL);
        for(j=0;j<i; j++)
        {
            smallest = j;
            for(k=j+1; k<i; k++)
            {
                if(sort[k].sproduct_price < sort[smallest].sproduct_price)
                {
                    smallest=k;
                }
                temp = sort[j];
                sort[j] = sort[smallest];
                sort[smallest] = temp;
            }
        }
        printf("\nSorted list for product price: \n");
        printf("\n-----------------------------------------------------------\n");
        for(int a=0; a<i; a++)
        {
            printf(" Product Number: %d \n Product Name: %s\n Product Price: RM%.2f\n Product
Quantity: %.f\n Product Tax: %.f\n Product Discount(%): %.f\n Product Net Amount: RM%.2f",
sort[a].sproduct_number, sort[a].sproduct_name, sort[a].sproduct_price, sort[a].sproduct_quantity,
sort[a].sproduct_tax, sort[a].sproduct_discount, sort[a].sproduct_net_amount);
            printf("\n-----------------------------------------------------------\n");
        }
    }
    getch();
}


void quantitylist(struct sorting sort[200], int i)
{
    int smallest;
    int j,k;
    struct sorting temp;
    if (productheadptr==NULL)
            {
                system("cls");
                        printf("\nEmpty list");
                        getch();
                        return productdetails();
            }
    else
    {
        system("cls");
        productcurrentptr=productheadptr;
```

```
        do
        {
           sort[i].sproduct_number = productcurrentptr->product_number;
           strcpy(sort[i].sproduct_name,productcurrentptr->product_name);
           sort[i].sproduct_price = productcurrentptr->product_price;
           sort[i].sproduct_quantity = productcurrentptr->product_quantity;
           sort[i].sproduct_tax = productcurrentptr->product_tax;
           sort[i].sproduct_discount = productcurrentptr->product_discount;
           sort[i].sproduct_net_amount = productcurrentptr->product_net_amount;
           productcurrentptr = productcurrentptr -> link;
           i++;
        }
        while(productcurrentptr != NULL);
        for(j=0;j<i; j++)
        {
           smallest = j;
           for(k=j+1; k<i; k++)
           {
              if(sort[k].sproduct_quantity < sort[smallest].sproduct_quantity)
              {
                 smallest=k;
              }
              temp = sort[j];
              sort[j] = sort[smallest];
              sort[smallest] = temp;
           }
        }
        printf("\nSorted list for product quantity: \n");
        printf("\n-----------------------------------------------------------\n");
        for(int a=0; a<i; a++)
        {
           printf("\ Product Number: %d\n Product Name: %s\n Product Price: RM%.2f\n Product
Quantity: %.f\n Product Tax: %.f\n Product Discount(%): %.f\n Product Net Amount: RM%.2f",
sort[a].sproduct_number, sort[a].sproduct_name, sort[a].sproduct_price, sort[a].sproduct_quantity,
sort[a].sproduct_tax, sort[a].sproduct_discount, sort[a].sproduct_net_amount);
           printf("\n-----------------------------------------------------------\n");
        }
     }
     getch();
}

void numberlist(struct sorting sort[200], int i)
{
     int smallest;
     int j,k;
     struct sorting temp;
     if (productheadptr==NULL)
           {
```

```
            system("cls");
        printf("\nEmpty list");
        getch();
        return productdetails();
         }
   else
   {
      system("cls");
      productcurrentptr=productheadptr;
      do
      {
         sort[i].sproduct_number = productcurrentptr->product_number;
         strcpy(sort[i].sproduct_name,productcurrentptr->product_name);
         sort[i].sproduct_price = productcurrentptr->product_price;
         sort[i].sproduct_quantity = productcurrentptr->product_quantity;
         sort[i].sproduct_tax = productcurrentptr->product_tax;
         sort[i].sproduct_discount = productcurrentptr->product_discount;
         sort[i].sproduct_net_amount = productcurrentptr->product_net_amount;
         productcurrentptr = productcurrentptr -> link;
         i++;
      }
      while(productcurrentptr != NULL);
      for(j=0;j<i; j++)
      {
         smallest = j;
         for(k=j+1; k<i; k++)
         {
            if(sort[k].sproduct_number < sort[smallest].sproduct_number)
            {
               smallest=k;
            }
            temp = sort[j];
            sort[j] = sort[smallest];
            sort[smallest] = temp;
         }
      }
      printf("\nSorted list for number product: \n");
      printf("\n-----------------------------------------------------------\n");
      for(int a=0; a<i; a++)
      {
         printf(" Product Number: %d \n Product Name: %s\n Product Price: RM%.2f\n Product
Quantity: %.f\n Product Tax: %.f\n Product Discount(%): %.f\n Product Net Amount: RM%.2f",
sort[a].sproduct_number, sort[a].sproduct_name, sort[a].sproduct_price, sort[a].sproduct_quantity,
sort[a].sproduct_tax, sort[a].sproduct_discount, sort[a].sproduct_net_amount);
         printf("\n-----------------------------------------------------------\n");
      }
   }
   getch();
```

```
}

float retnetamt()
{
   return(productcurrentptr->product_net_amount);
}

void reportbill()
{
   float gtotal;
   if (productheadptr==NULL)
          {
             system("cls");
                   printf("\nEmpty list");
                   getch();
                   return;
          }
          productcurrentptr = productheadptr;
   system("cls");

   gotoxy(30,3);
   printf(" PRODUCT DETAILS ");
   gotoxy(3,5);
   printf("NUMBER");
   gotoxy(13,5);
   printf("NAME");
   gotoxy(23,5);
   printf("PRICE");
   gotoxy(33,5);
   printf("QUANTITY");
   gotoxy(44,5);
   printf("TAX");
   gotoxy(52,5);
   printf("DISCOUNT");
   gotoxy(64,5);
   printf("NET AMOUNT");
   int k=7;
   do
   {
      gotoxy(3,k);
      printf("%d", productcurrentptr->product_number);
      gotoxy(13,k);
      printf("%s", productcurrentptr->product_name);
      gotoxy(23,k);
      printf("RM%.2f",productcurrentptr->product_price);
      gotoxy(33,k);
      printf("%.f",productcurrentptr->product_quantity);
      gotoxy(44,k);
```

```
        printf("%.f",productcurrentptr->product_tax);
        gotoxy(52,k);
        printf("%.f",productcurrentptr->product_discount);
        gotoxy(64,k);
        printf("%.f",productcurrentptr->product_net_amount);
        k=k+1;
        gtotal = gtotal + retnetamt();
                        productcurrentptr=productcurrentptr->link;
            }
            while(productcurrentptr != NULL);
            printf("\nGrand Total = RM%.2f",gtotal);
            getch();
            return;
}

void updateproduct(struct sorting sort[200])
{
   system("cls");
   int number,ans,i=0;
   numberlist(sort,i);
   printf("\nPRESS ENTER TO CONTINUE RUNNING.....");
   productcurrentptr = productheadptr;
   if (productcurrentptr == NULL)
   {
      printf("\nEmpty List");
      getch();
      return productdetails();
   }
   else
   {
      system("cls");
      printf("Enter product number that you want to update: ");
      scanf("%d", &number);

      ans = binary_search(sort,i,number);
      printf("\nEnter product price: ");
      scanf("%f",&sort[ans].sproduct_price);
      printf("\nEnter product quantity: ");
      scanf("%f",&sort[ans].sproduct_quantity);
      printf("\nEnter tax percent: ");
      scanf("%f",&sort[ans].sproduct_tax);
      printf("\nEnter product discount: ");
      scanf("%f",&sort[ans].sproduct_discount);
      sort[ans].sproduct_gross =
sort[ans].sproduct_price+(sort[ans].sproduct_price*(sort[ans].sproduct_tax/100));
      sort[ans].sproduct_net_amount =
sort[ans].sproduct_quantity*(sort[ans].sproduct_gross-(sort[ans].sproduct_gross*(sort[ans].sproduct_
discount/100)));
```

```
        if(productcurrentptr->product_number == number && productcurrentptr != NULL)
            {
                productcurrentptr->product_price = sort[ans].sproduct_price;
                productcurrentptr->product_quantity = sort[ans].sproduct_quantity;
                productcurrentptr->product_tax = sort[ans].sproduct_tax;
                productcurrentptr->product_discount = sort[ans].sproduct_discount;
                productcurrentptr->product_gross = sort[ans].sproduct_gross;
                productcurrentptr->product_net_amount = sort[ans].sproduct_net_amount;
            }
        else
        {
            productcurrentptr= productcurrentptr->link;
        }
    }

    printf("\n\nSUCCESSFULLY UPDATED!!");
    getch();
    return productdetails();
}

void binary_search(struct sorting sort[200], int i, int number)
{
    int first=0;
    int last=i-1;
    int mid;
    while(first<=last)
    {
        mid=(first+last)/2;
        printf("\nValue for mid: %d",mid);

        if (number==sort[mid].sproduct_number)
        {
            printf("\nproduct Number : %d\n", sort[mid].sproduct_number);
            printf("product Name: %s\n", sort[mid].sproduct_name);
            printf("\n-----------------------------------------\n");
            return mid;
        }
        else
        {
            if(number<sort[mid].sproduct_number)
                last =mid-1;
            else
                first=mid+1;
        }
    }
}

void quit()
```

```
{
   system("cls");

   printf("ARE YOU SURE, YOU WANT TO EXIT (Y/N)?");
   char yn;
   scanf(" %c",&yn);
   if((yn=='Y')||(yn=='y'))
   {

      system("cls");
      printf("THANKS FOR USING THIS SYSTEM!!");
      getch();
      exit(0);
   }
   else if((yn=='N')||(yn=='n'))
   return;
   else
   {
      printf("INVALID ANSWER, YOU WILL RETURN BACK TO MAIN MENU...");
      return;
   }
}

int main()
{
   int p=TRUE;
   int op;
         productheadptr = NULL;
         customerheadptr = NULL;
   while (p==TRUE)//13 is enter key of c
   {
      system("cls");// Clearing Screen
      printf("\t\t%c",201);//print left top corner from ASCII
      for(int i=0;i<26;i++)
      {
         printf("%c",205);sleep(6000);
      }
      printf("%c\n",187);//print top right corner from ASCII
      printf("\t\t%cSUPERMARKET BILLING SYSTEM",186);//print vertical borders from ASCII
      printf("%c\n",186);
      printf("\t\t%c",200);//print bottom left corner from ASCII
      for(int i=0;i<26;i++)
      {
         printf("%c",205);sleep(6000);
      }
      printf("%c\n",188);//print bottom right corner from ASCII
      printf("\n-------------------------------------------------\n");
      printf("[1] - CUSTOMER DETAILS \n");
```

```
        printf("[2] - PRODUCT DETAILS \n");
        printf("[3] - REPORT BILL OF PRODUCTS\n");
        printf("[4] - EXIT \n");//goes to x,y console);
        printf("\n-------------------------------------------------\n");
        printf("Which option you want to choose? ");
        scanf("%d",&op);
        switch (op)
        {
           case 1 :customerdetails();break;
           case 2 :productdetails();break;
           case 3 :reportbill();break;
           case 4 :quit(); break;
           default:printf("\nWrong Option....Please Retry!");getch();return main();
        }
    }
}
```

3.0 REFERENCE

Lotus's: Shop Conveniently & Get Rewarded with Lotus's. Malaysia. (n.d.). Retrieved January 10, 2023, from https://www.lotuss.com.my/en

Supermarket billing system. Nevon Projects. (2018, November 26). Retrieved January 10, 2023, from https://nevonprojects.com/supermarket-billing-system/#:~:text=The%20supermarket%20billing%20system%20is,providing%20an%20efficient%20customer%20service

Powerful supermarket billing software to automate your supermarket store in India. Best Supermarket Billing Software to Manage Your Supermarket. (n.d.). Retrieved January 9, 2023, from https://margcompusoft.com/retail/supermarket_software.html