**ITC Group Assignment**

**Group member details:**

John Adriaans 220038627

Hannah Dunaiski 220086915

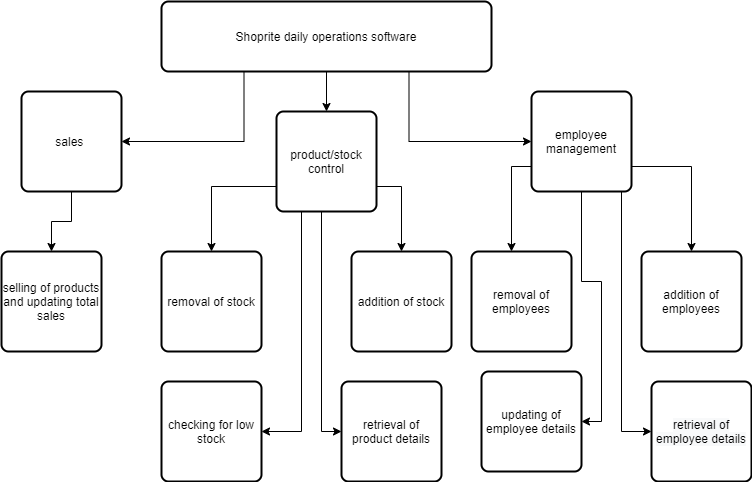
Revaldo Gertze 220056676

Uahengisa Katjizuko 220040850

Shaun Kamesiepo 220053685

Github repo: <https://github.com/John-Ad/ITC-group-project>

**Modules**

****

**Functions**

//---------beginning of program--------------------------------------

START

Class employee //stores employee inFORmation

{

var id;

var name;

var age;

var position;

var hours\_worked;

var employment\_type;

var dollars\_per\_hour;

var wages;

}

Class product //stores product information

{

var id;

var name;

var price;

var amount;

}

employee emp[]; //array of employees that can increase or decrease in size

product prod[]; //array of products that can increase or decrease in size

var emp\_count=0; //keeps track of number of employees

var prod\_count=0; //keeps track of number of products

var running==true; //states whether or not the program is running

var sales\_for\_month=0;

CALL main(); //calls the main function

END

//---------end of program-------------------------------------------

//main function:

FUNCTION main

PASS IN: nothing //no arguments are passed to the main function

var event=NULL; //this variable represents an action within the program

var id,name,age,position,salary,emp\_type,hrs\_worked,d\_p\_h;

employee emp\_; //creates an object of the type employee

product prod\_; //creates an object of the type product

WHILE(running==true) //when this loop ends, the program will exit

IF(program is exited) //IF exit button is clicked or another exit command is given

running=false;

ENDIF

prod\_ = CALL check\_low\_stock(); //check\_low\_stock returns an object of the type product or NULL IF no product is //running low

IF(prod\_!=NULL)

DISPLAY(prod\_.name + " is low and needs to be ordered!"); //IF a product is running low, it will be displayed

ENDIF

READ event; // IF a user performs an action, that action will be stored in event

SWITCH(event) //switch compares event with the following cases

CASE products sold:

CALL products\_sold(); //processes the transaction

CASE employee hired:

READ name,age,position,d\_p\_h,emp\_type;

CALL add\_emp(arguments: name,age,position,d\_p\_h,emp\_type); //adds a new employee

CASE employee fired/quit:

READ id;

CALL remove\_emp(arguments: id); //removes an employee

CASE product details requested:

READ id;

prod\_ = CALL get\_prod\_details(arguments: id); //assigns an object of type product to prod

DISPLAY("name: " + prod\_.name + " price: " + prod\_.price + " amount in stock: " + prod\_.amount); //displays product

//info

CASE stock received:

READ id;

READ amount;

CALL add\_stock(arguments: id,amount);

CASE employee details requested:

READ id;

emp\_ = CALL get\_emp\_details(arguments: id); //assigns an object of type employee to emp

DISPLAY("name: " + emp\_.name + " age: " + emp\_.age + " position: "emp\_.position + " hours worked: " emp\_.hours\_worked + " employment type: " + emp\_.employment\_type); //displays employee info

CASE employee details updated:

READ id,name,age,position,hrs\_worked,emp\_type

CALL update\_emp\_details(arguments: id, name, age, position, hrs\_worked, emp\_type);

CASE calculate wage for all employees:

CALL calculate\_wage();

CASE get sales\_for\_month:

DISPLAY(sales\_for\_month);

ENDSWITCH

prod\_=NULL; //prod\_ is emptied so that low stock can be checked on the next iteration of the loop

ENDWHILE

PASS OUT: nothing //the main function does not return anything

ENDFUNCTION

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

//products sold

FUNCTION products\_sold

PASS IN: nothing

var id=0;

var quantity=0;

var total=0;

var more\_items=true;

var ans="";

while(more\_items==true)

{

DISPLAY("Enter product id: ");

READ id;

DISPLAY("Enter quantity: ");

READ quantity;

for( var i= 0 to prod\_count-1 )

{

if(prod[i].id==id)

{

total+=prod[i].price\*quantity;

Call remove\_stock(arguments: id, quantity);

i=prod\_count;

}

}

DISPLAY("Are there more items Y or N: ")

READ ans;

if(ans=='N')

{

more\_items=false;

}

}

sales\_for\_month+=total;

DISPLAY("Total cost: N$"+total);

PASS OUT: nothing

END FUNCTION

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

//return product/employee details

FUNCTION get\_prod\_details

PASS IN: id

FOR( var i= 0 to prod\_count-1)

IF(prod[i].id==id)

PASS OUT: prod[i]; //returns the variable that contains the employee's data

ENDIF

ENDFOR

PASS OUT: nothing //returns nothing if no employee was found

ENDFUNCTION

FUNCTION get\_emp\_details

PASS IN: id

FOR( var i= 0 to emp\_count-1)

IF(emp[i].id==id)

PASS OUT: emp[i];

i=emp\_count;

ENDIF

ENDFOR

PASS OUT: nothing

ENDFUNCTION

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

//removal/addition of stock

FUNCTION remove\_stock

PASS IN: id, amount

FOR( var i= 0 to prod\_count-1) // loops through all the products

IF(prod[i].id==id) // finds the product with the matching id

prod[i].amount-=amount;

i=prod\_count;

ENDIF

ENDFOR

PASS OUT: nothing

ENDFUNCTION

FUNCTION add\_stock

PASS IN: id, amount

FOR(i= 0 to prod\_count-1) // loops through all the products

IF(prod[i].id==id) // finds the product with the matching id

prod[i].amount+=amount;

ENDIF

ENDFOR

PASS OUT: nothing

ENDFUNCTION

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

//add/remove employee

FUNCTION add\_emp

PASS IN: name,age,position,employment\_type,dollars\_per\_hour

increase emp size by 1; //creates space for new employee

increase emp\_count by 1;

emp[emp\_count-1].id=emp\_count;

emp[emp\_count-1].name=name;

emp[emp\_count-1].age=age;

emp[emp\_count-1].position=position;

emp[emp\_count-1].hours\_worked=0;

emp[emp\_count-1].employment\_type=employment\_type;

emp[emp\_count-1].dollars\_per\_hour=dollars\_per\_hour;

emp[emp\_count-1].wage=0;

PASS OUT: nothing

ENDFUNCTION

FUNCTION remove\_emp

PASS IN: id

FOR( var i= 0 to emp\_count-1)

IF(emp[i].id==id)

delete emp[i];

decrease emp size by 1;

decrease emp\_count by 1;

i=emp\_count; //ends loop

ENDIF

ENDFOR

PASS OUT: nothing

ENDFUNCTION

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

//update employee details

FUNCTION update\_emp\_details

PASS IN: id, name, age, position, hours\_worked, employment\_type

FOR(int i= 0 to emp\_count-1)

IF(emp[i].id==id) //finds the matching employee

IF(name!="") // each of variables will be checked to see if they are empty

Emp[i].name=name; // employers might only want to change 1 or 2 details, therefore, some arguments in // the function call can be left empty

ENDIF // if the variable is not empty, its value will be assigned to the matching employee variable

IF(age!="")

Emp[i].age=age;

ENDIF

IF(position!="")

Emp[i].position=position;

ENDIF

IF(hours\_worked!="")

Emp[i].hours\_worked=hours\_worked;

ENDIF

IF(employment\_type!="")

Emp[i].employment\_type=employment\_type;

ENDIF

ENDIF

ENDFOR

PASS OUT: nothing

ENDFUNCTION

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

//calculate wages for all employees

FUNCTION calculate\_wage

PASS IN: nothing

var wage=0;

FOR(int i= 0 to emp\_count-1)

emp[i].wage = emp[i].hours\_worked \* emp[i].dollars\_per\_hour;

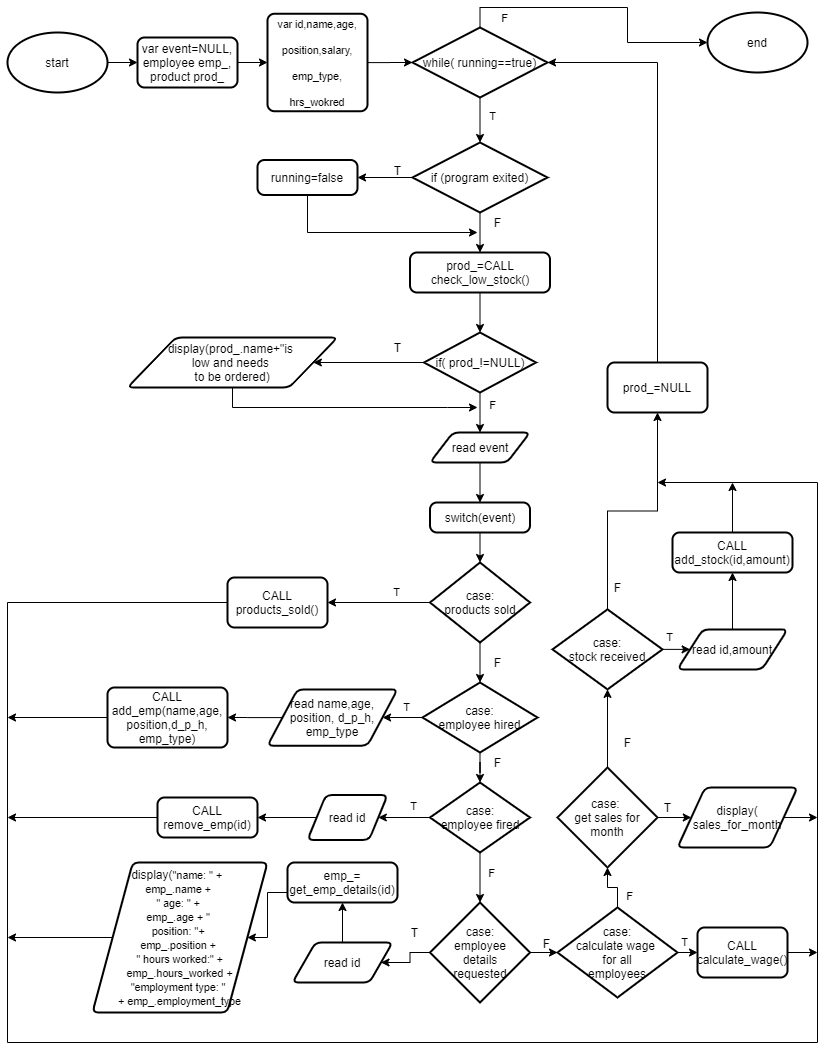
ENDFOR

PASS OUT: nothing

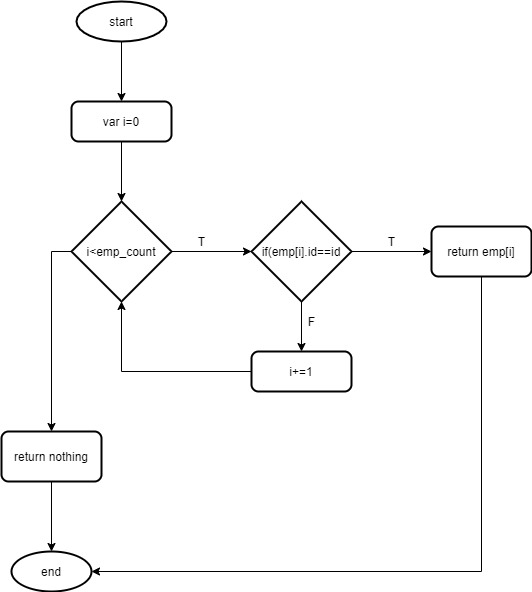
ENDFUNCTION

**Flowcharts**: start point to end point of program:

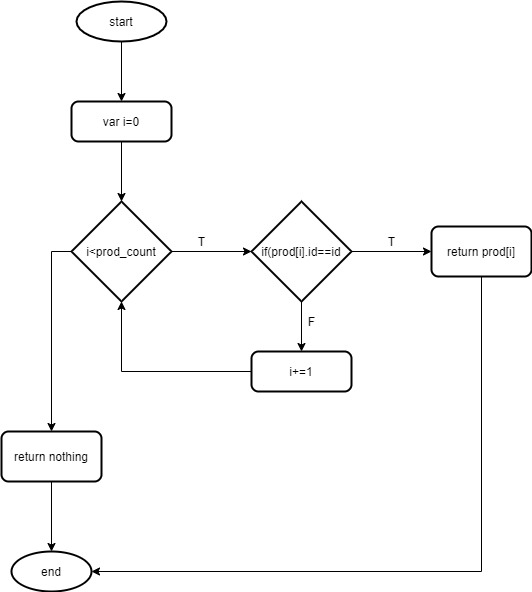


main():

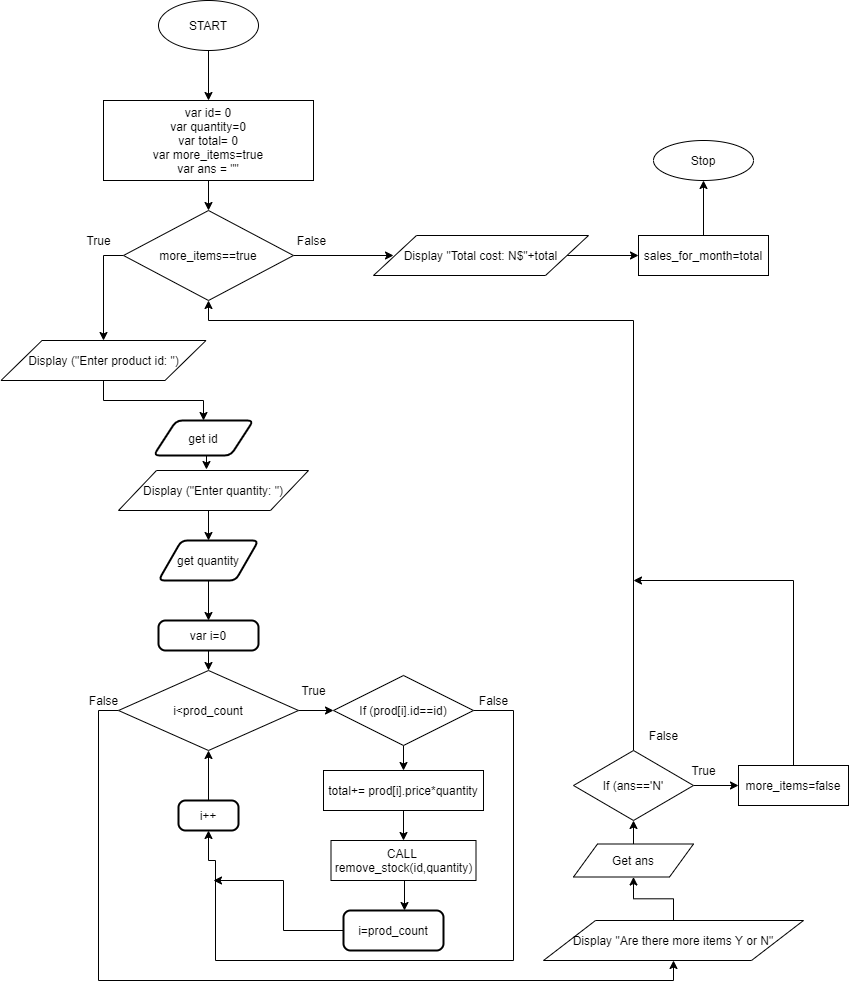
get\_emp\_details(var id):

****

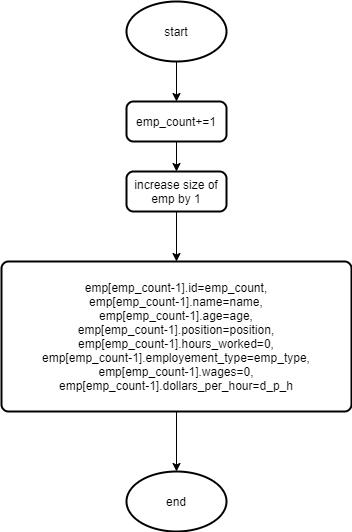
get\_prod\_details(var id):

****

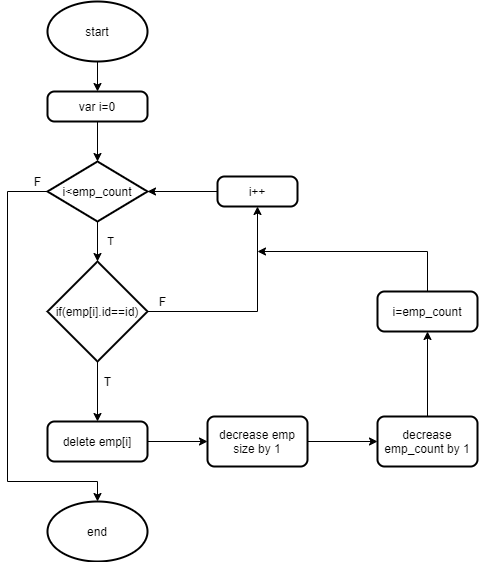
products\_sold():



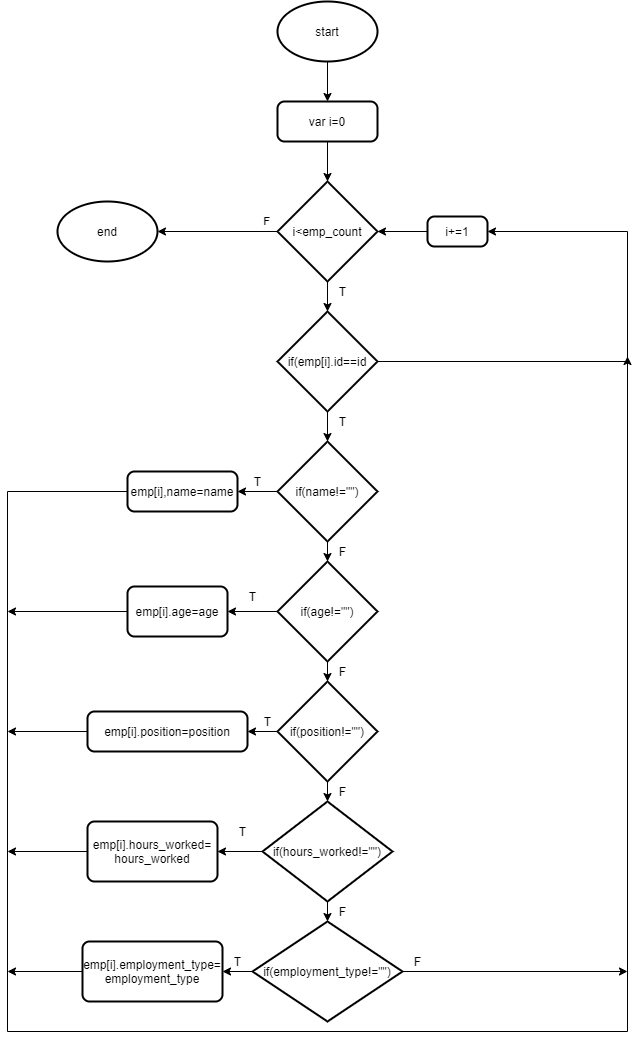
Add\_emp(var name,age,position,d\_p\_h,emp\_type):



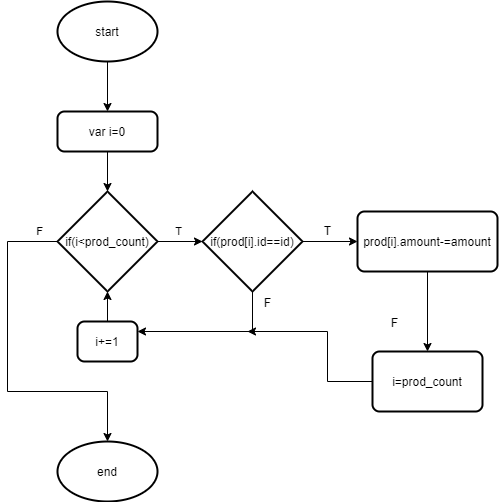
remove\_emp(var id):



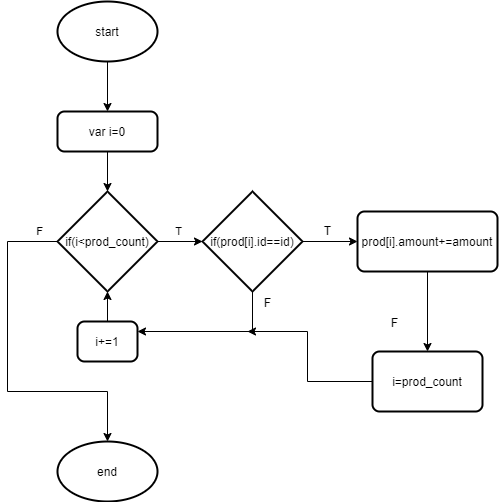
update\_emp\_details(var id,name):,age,hrs\_worked,position):



remove\_stock(var id):



add\_stock(var id):



calculate\_wage():

