DHRUV GARG 16BCE1190

Programme	:	**B.Tech (ECE|CSE)**	Semester	:	**FS 2017-18**
Course	:	**DATA STRUCTURES AND ALGORITHMS**	Code	:	**CSE2003**
Faculty	:	**Dr.Vetrivelan.P**	Slot	:	**G1**

Digital Assignment-#1 Questions
(Implementation in C Programs)

1. **Create a Restaurant using doubly linked-list (circular type) and perform the following operations with an appropriate pseudo-code:**
   - **(i) Create an order**
   - **(ii) Insert an order in middle**
   - **(iii) Delete one of the order (before prepared)**
   - **(iv) Search an order for Juice category**
   - **(v) Display the bill with ordered items**

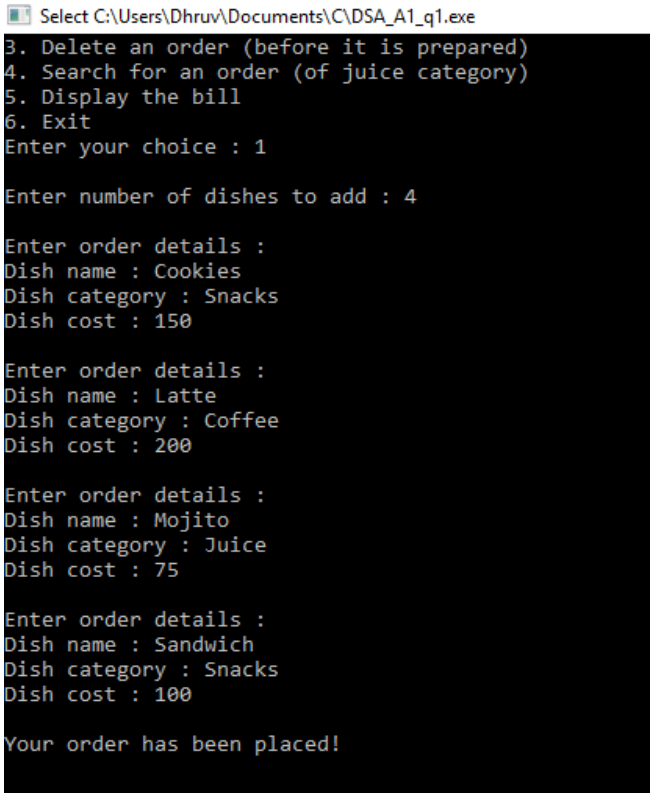
CODE :

```
# include <stdio.h>
# include <stdlib.h>
# include <conio.h>
struct cdlinklist
{
   struct cdlinklist *left;
   char name[20];
   char category[10];
   int cost;
   struct cdlinklist *right;
};
typedef struct cdlinklist node;
node *start = NULL;
int nodectr;
int totalBill = 0;
node* getnode()
{
   node * newnode;
   newnode = (node *) malloc(sizeof(node));
   printf("\nEnter order details : ");
   printf("\nDish name : ");
   scanf("%s", newnode->name);
   printf("Dish category : ");
   scanf("%s", newnode->category);
   printf("Dish cost : ");
   scanf("%d", &newnode->cost);
   newnode -> left = NULL;
   newnode -> right = NULL;
   return newnode;
}
int menu()
{
   int ch;
   printf("\n**** STARBUCKS COFFEE, CHENNAI
****");
   printf("\n1. Create ");
   printf("\n2. Insert an order in the middle");
   printf("\n3. Delete an order (before it is
prepared)");
   printf("\n4. Search for an order (of juice category)");
   printf("\n5. Display the bill");
   printf("\n6. Exit");
   printf("\nEnter your choice : ");
   scanf("%d", &ch);
   return ch;
}
void cdll_createlist(int n)
{
   int i;
   node *newnode, *temp;
   if(start == NULL)
   {
      nodectr = n;
      for(i = 0; i < n; i++)
      {
         newnode = getnode();
         if(start == NULL)
         {
            start = newnode;
            newnode -> left = start;
            newnode ->right = start;
         }
         else
         {
            newnode -> left = start -> left;
            newnode -> right = start;
            start -> left->right = newnode;
```

```
            start -> left = newnode;
          }
       }
    }
    else
       printf("\n List already exists..");
}
void cdll_display_left_right()
{
    node *temp;
    temp = start;
    if(start == NULL)
       printf("\n Empty List");
    else
    {
       printf("\n \t ORDER INFORMATION \n");
       printf("\n----------------------------------------\n");
       printf("\nName\t\tCategory\tPrice\n");
       printf("%s\t\t", temp -> name);
       printf("%s\t\t", temp -> category);
       printf("%d", temp -> cost);
       totalBill += temp->cost;
       temp = temp -> right;
       while(temp != start)
       {
          printf("\n%s\t\t", temp -> name);
          printf("%s\t\t", temp -> category);
          printf("%d", temp -> cost);
          totalBill += temp->cost;
          temp = temp -> right;
       }
    }
    printf("\n----------------------------------------\n");
    printf("\tTOTAL BILL AMOUNT IS : %d", totalBill);
    printf("\n----------------------------------------\n");
    printf("\n");
}

void cdll_insert_mid()
{
    node *newnode, *temp, *prev;
    int pos, ctr = 1;
    newnode = getnode();
    printf("\nEnter the position: ");
    scanf("%d", &pos);
    if(pos - nodectr >= 2)
    {
       printf("\nPosition is out of range.");
       return;
    }
    if(pos > 1 && pos <= nodectr)
    {
       temp = start;
       while(ctr < pos - 1)
       {
          temp = temp -> right;
          ctr++;
       }
       newnode -> left = temp;
       newnode -> right = temp -> right;
       temp -> right -> left = newnode;
       temp -> right = newnode;
       nodectr++;
       printf("\nOrder inserted in the middle.\n");
    }
    else
       printf("\nPosition %d of list is not a middle
position", pos);
}

void search(){
            printf("\nEnter category of items to be
searched : ");
            char categ[10];
    scanf("%s", &categ);
            node *temp = start;
            int z=0;
            do{
                        if(!strcmp(temp->category,categ)){
             if(z == 0)
             {
                printf("Item found!\n");
                printf("\n----------------------------------------
\n");
                printf("\nName\t\tCategory\t\tPrice");
                printf("\n----------------------------------------
\n");
             }
                                    printf("\n%s\t\t%s\t\t%d
",temp->name,temp->category,temp->cost);
                                    z=1;
                        }
       temp = temp->right;
            }while(temp!=start);
            if(z==0)
                        printf("\nSorry, order not found.");
    printf("\n");
}

void cdll_delete_last()
{
    node *temp;
    if(start == NULL)
    {
       printf("\nNo nodes exist.");
       getch();
       return;
    }
    else
    {
```

```
      nodectr--;
      if(nodectr == 0)
      {
         free(start);
         start = NULL;
      }
      else
      {
         temp = start;
         while(temp -> right != start)
            temp = temp -> right;
         printf("\nThe order being deleted is the last
order to be placed : \n");
         printf("\n----------------------------------------");
         printf("\nName\t\tCategory\t\tPrice");
         printf("\n----------------------------------------\n");
         printf("\n%s\t\t%s\t\t%d \n",temp-
>name,temp->category,temp->cost);
         temp -> left -> right = temp -> right;
         temp -> right -> left = temp -> left;
         free(temp);
      }
      printf("\nORDER DELETED since it was not
prepared.\n");
   }
}

void main(void)
{
   int ch,n;
   while(1)
   {
      ch = menu();
      switch( ch)
      {
      case 1 :
         printf("\nEnter number of dishes to add : ");
         scanf("%d", &n);
         cdll_createlist(n);
         printf("\nYour order has been placed! \n");
         break;
      case 2 :
         cdll_insert_mid();
         break;
      case 3 :
         cdll_delete_last();
         break;
      case 4 :
         search();
         break;
      case 5 :
         totalBill = 0;
         cdll_display_left_right();
         break;
      case 6:
         exit(0);
      }
      getch();
   }
}
```

Creating an order



Inserting an order in the middle

```
**** STARBUCKS COFFEE, CHENNAI ****
1. Create
2. Insert an order in the middle
3. Delete an order (before it is prepared)
4. Search for an order (of juice category)
5. Display the bill
6. Exit
Enter your choice : 2

Enter order details :
Dish name : Cocktail
Dish category : Juice
Dish cost : 120

Enter the position: 2

Order inserted in the middle.
```

Searching an order for Juice category

```
**** STARBUCKS COFFEE, CHENNAI ****
1. Create
2. Insert an order in the middle
3. Delete an order (before it is prepared)
4. Search for an order (of juice category)
5. Display the bill
6. Exit
Enter your choice : 4

Enter category of items to be searched : Juice
Item found!

----------------------------------------

Name            Category                Price
----------------------------------------

Cocktail                Juice           120
SB_Mojito               Juice           75
```

Display the bill with ordered items

```
**** STARBUCKS COFFEE, CHENNAI ****
1. Create
2. Insert an order in the middle
3. Delete an order (before it is prepared)
4. Search for an order (of juice category)
5. Display the bill
6. Exit
Enter your choice : 5

        ORDER INFORMATION

----------------------------------------

Name            Category        Price
SB_Cookie               Snacks          150
Cocktail                Juice           120
SB_Latte                Coffee          200
SB_Mojito               Juice           75
Sandwich                Snacks          100
----------------------------------------
       TOTAL BILL AMOUNT IS : 645
----------------------------------------
```

Deleting one of the (order before it is prepared)

```
**** STARBUCKS COFFEE, CHENNAI ****
1. Create
2. Insert an order in the middle
3. Delete an order (before it is prepared)
4. Search for an order (of juice category)
5. Display the bill
6. Exit
Enter your choice : 3

The order being deleted is the last order to be placed :

----------------------------------------
Name            Category                Price
----------------------------------------

Sandwich                Snacks          100

ORDER DELETED since it was not prepared.
```

Displaying the updated bill

```
**** STARBUCKS COFFEE, CHENNAI ****
1. Create
2. Insert an order in the middle
3. Delete an order (before it is prepared)
4. Search for an order (of juice category)
5. Display the bill
6. Exit
Enter your choice : 5

         ORDER INFORMATION

----------------------------------------

Name            Category        Price
SB_Cookie               Snacks          150
Cocktail                Juice           120
SB_Latte                Coffee          200
SB_Mojito               Juice           75
----------------------------------------
        TOTAL BILL AMOUNT IS : 545
----------------------------------------
```

2. **An automated system is designed for palindrome check which has equipped with LIFO based access memory. The system reads one character at a time from the input symbol which contains both word (W) and reversal of word (WR). Sample case, W#W^R: 110#011**
   (i) Push into LIFO memory (one by one before this symbol "#")
   (ii) Do not perform either insert or delete if you read this symbol "#"
   (iii) Pop from LIFO memory (after this symbol "#") only if the read symbol is same as the top element
   After ending the reading, if LIFO memory is empty then print "PALINDROME". Otherwise, print "NON-PALINDROME".

CODE :

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
struct stack
{
   int data;
   struct stack *next;
};
void push();
void pop();
void display();
typedef struct stack node;
node *start=NULL;
node *top = NULL;
node* getnode()
{
   node *temp;
   temp=(node *) malloc( sizeof(node)) ;
   printf("Enter data : ");
   scanf("%d", &temp -> data);
   temp -> next = NULL;
   return temp;
}
void push(node *newnode)
{
   node *temp;
   if( newnode == NULL )
   {
      printf("Stack overflow! \n");
      return;
   }
   if(start == NULL)
   {
      start = newnode;
      top = newnode;
   }
   else
   {
      temp = start;
      while( temp -> next != NULL)
         temp = temp -> next;
      temp -> next = newnode;
      top = newnode;
   }
   printf("Data pushed into stack. \n");
}
int topmostEle()
{
   node *temp;
   if(top == NULL)
   {
      printf("Stack underflow! \n");
      return 0;
   }
   temp = start;
   if( start -> next == NULL)
   {
      return top -> data;
   }
   else
   {
      while(temp -> next != top)
      {
         temp = temp -> next;
      }
      return top -> data;
   }
}
void pop()
{
   node *temp;
   if(top == NULL)
   {
      printf("Stack underflow! \n");
      return;
   }
   temp = start;
   if( start -> next == NULL)
   {
      printf("Popped element is %d \n", top -> data);
      start = NULL;
      free(top);
      top = NULL;
   }
   else
   {
      while(temp -> next != top)
      {
```

```
        temp = temp -> next;
      }
      temp -> next = NULL;
      printf("Popped element is %d \n", top -> data);
      free(top);
      top = temp;
   }
}
void display()
{
   node *temp;
   if(top == NULL)
   {
      printf("Stack is empty! \n");
   }
   else
   {
      temp = start;
      printf("Elements in the stack: ");
      printf("%5d ", temp -> data);
      while(temp != top)
      {
         temp = temp -> next;
         printf("%5d ", temp -> data);
      }
      printf("\n");
   }
}
int count()
{
   node *temp;
   int s = 0;
   if(top == NULL)
   {
      s = 0;
   }
   else
   {
      temp = start;
      while(temp != top)
      {
         temp = temp -> next;
         s++;
      }
   }
   return s+1;
}
char menu()
{
   char ch;
   printf("\n\tSTACK OPERATIONS \n ");
   printf("-----------------------------\n");
   printf("1. Push \n");
   printf("2. Pop \n");
   printf("3. Display \n");
   printf("#. Stop entering numbers \n");
   printf("Enter your choice: ");
   ch = getchar();
   return ch;
}
void main()
{
   char ch;
   int size;
   node *newnode;
   do
   {
      ch = menu();
      switch(ch)
      {
      case '1' :
         newnode = getnode();
         push(newnode);
         break;
      case '2' :
         pop();
         break;
      case '3' :
         display();
         break;
      case '#':
         break;
      }
      getchar();
   }
   while( ch != '#' );

   size = count();
   int i = 0,ich;
   printf("\nEnter the characters to check for
palindrome : (Press # to exit)");
   printf("\n");
   do
   {
      printf("Enter the element : ");
      scanf("%[^\n]%*c", &ch);
      if(ch != '#')
      {
         ich = ch - '0';
         if(ich == topmostEle())
         {
            i++;
            printf("Entered element EQUALS peek!\n");
            pop();
         }
         else
         {
            printf("Entered element DOES NOT EQUAL
peek!\n\n");
            continue;
```

```
        }
        printf("\n");
    }
  }
  while(ch != '#');
  if(i == size)
  {
        printf("\nPALINDROME!\n");

    }
    else
        printf("\nNOT PALNDROME!\n");
}
```

Test case 1 : 110#011

Inserting "110" into the stack and printing is elements.

```
C:\Users\Dhruv\Documents\C\DSA_A1_q2.exe

        STACK OPERATIONS
 -----------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 1
Enter data : 1
Data pushed into stack.

        STACK OPERATIONS
 -----------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 1
Enter data : 1
Data pushed into stack.

        STACK OPERATIONS
 -----------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 1
Enter data : 0
Data pushed into stack.

        STACK OPERATIONS
 -----------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 3
Elements in the stack:      1      1      0
```

Inserting the reversed word to check for palindrome

C:\Users\Dhruv\Documents\C\DSA_A1_q2.exe

```
        STACK OPERATIONS
 ------------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: #

Enter the characters to check for palindrome : (Press # to exit)
Enter the element : 0
Entered element EQUALS peek!
Popped element is 0

Enter the element : 1
Entered element EQUALS peek!
Popped element is 1

Enter the element : 1
Entered element EQUALS peek!
Popped element is 1

Enter the element : #

PALINDROME!
```

Test case 2 : 1190#1190

Using my registration number 1190 as input for the list as well as the reversed word.

C:\Users\Dhruv\Documents\C\DSA_A1_q2.exe

```
        STACK OPERATIONS
 ------------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 1
Enter data : 1
Data pushed into stack.

        STACK OPERATIONS
 ------------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 1
Enter data : 1
Data pushed into stack.

        STACK OPERATIONS
 ------------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 1
Enter data : 9
Data pushed into stack.

        STACK OPERATIONS
 ------------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 1
Enter data : 0
Data pushed into stack.
```

Printing the stack contents and stopping further data entry into the stack.

```
        STACK OPERATIONS
 -----------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: 3
Elements in the stack:     1     1     9     0

        STACK OPERATIONS
 -----------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: #
```

Entering 1190 as WR to check for palindrome. Pop occurs when peek = input. Enter # to stop further input.

C:\Users\Dhruv\Documents\C\DSA_A1_q2.exe

```
        STACK OPERATIONS
 -----------------------------
1. Push
2. Pop
3. Display
#. Stop entering numbers
Enter your choice: #

Enter the characters to check for palindrome : (Press # to exit)
Enter the element : 1
Entered element DOES NOT EQUAL peek!

Enter the element : 1
Entered element DOES NOT EQUAL peek!

Enter the element : 9
Entered element DOES NOT EQUAL peek!

Enter the element : 0
Entered element EQUALS peek!
Popped element is 0

Enter the element : #

NOT PALNDROME!
```

3. **Create a Binary Tree and perform the following:**
   - **(i) Create**
   - **(ii) Insert**
   - **(iii) Delete of node**
   - **(iv) Display**
   - **(v) Count**

CODE :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef struct bt
{
   int data;
   struct bt *left,*right;
} node;

node *create(node *);
void insert(node *,node *);
void display(node *);
void inorder(node *);
void preorder(node*);
void postorder(node *);
void count(node *);
node *deleteNode(node *, int);
node *minValueNode(node *);
node *get_node();
node *root;
int c = 0;

void main()
{
   int choice,val;
   node *tmp,*parent;
   root=NULL,parent=NULL;
   printf("\n\tREPRESENTATION OF BINARY TREE");
   printf("\n1. Create\n2. Insert \n3. Delete a node
\n4. Display \n5. Count\n6. Exit");
   while(1)
   {
      printf("\nEnter your choice : ");
      scanf("%d",&choice);
      switch(choice)
      {
      case 1:
         root=create(root);
         break;
      case 2:
         create(root);
         break;
      case 3 :
         {int key;
         printf("Enter the value you wish to delete : ");
         scanf("%d", &key);
         deleteNode(root,key);
         printf("The element %d has been deleted.\n",
key);
         break;}
      case 4 :
         display(root);
         break;
      case 5 :
         c = 0;
         count(root);
         printf("Number of elements in the tree : %d\n",
c);
         break;
      default:
         printf("END OF PROGRAM\n");
         exit(0);
      }
   }
   getch();
}

node *create(node *root)
{
   node *New;
   int val;
   char ans;
   do
   {
      printf("\nEnter the element : ");
      scanf("%d",&val);
      New=get_node();
      if(New==NULL)
      {
         printf("\nMemory is not allocated.");
         return;
      }
      New->data=val;
      if(root==NULL)
         root=New;
      else
         insert(root,New);
      printf("Do you want to continue (y/n) : ");
```

```
      ans=getche();
      printf("\n");
   }
   while((ans=='Y')||(ans=='y'));
   printf("\nBinary tree has been created.\n");
   return(root);
}

node *get_node()
{
   node *temp;
   temp=(node*)malloc(sizeof(node));
   temp->left=NULL;
   temp->right=NULL;
   return(temp);

}

void insert(node *root,node *New)
{
   char ch;
   printf("Where to insert l/r of %d ? : ",root->data);
   ch=getche();
   printf("\n");
   if((ch=='r')||(ch=='R'))
   {
      if(root->right==NULL)
         root->right=New;
      else
         insert(root->right,New);
   }
   else if((ch=='l')||(ch=='L'))
   {
      if(root->left==NULL)
         root->left=New;
      else
         insert(root->left,New);
   }
   else
      printf("\nEntered wrong choice");
}

void display(node *temp)
{
   int ans;
   printf("Displaying in inorder : \n");
   inorder(temp);
   printf("\n");
}

void inorder(node *temp)
{
   if(temp!=NULL)
   {
      inorder(temp->left);
      printf(" %d",temp->data);
      inorder(temp->right);
   }
}

void count(node *temp)
{
   if(temp!=NULL)
   {
      count(temp->left);
      c++;
      count(temp->right);
   }
}

node *minValueNode(node *temp)
{
   while(temp->left != NULL)
      temp = temp -> left;
   return temp;
}

node *deleteNode(node *root, int data)
{
   if(root == NULL)
      return root;
   if(data < root->data)
      root->left = deleteNode(root->left, data);
   else if(data > root->data)
      root->right = deleteNode(root->right, data);
   else
   {
      if(root -> left == NULL)
      {
         node *temp = root->right;
         free(root);
         return temp;
      }
      else if(root->right == NULL)
      {
         node *temp = root -> left;
         free(root);
         return temp;
      }
      node *temp = minValueNode(root->right);
      root->data = temp->data;
      root->right = deleteNode(root->right, temp-
>data);
   }
   return root;
}
```

DHRUV GARG 16BCE1190

Creation of Binary tree

C:\Users\Dhruv\Documents\C\DSA_A1_q3.exe

```
        REPRESENTATION OF BINARY TREE
1. Create
2. Insert
3. Delete a node
4. Display
5. Count
6. Exit
Enter your choice : 1

Enter the element : 60
Do you want to continue (y/n) : y

Enter the element : 32
Where to insert l/r of 60 ? : l
Do you want to continue (y/n) : y

Enter the element : 73
Where to insert l/r of 60 ? : r
Do you want to continue (y/n) : y

Enter the element : 12
Where to insert l/r of 60 ? : l
Where to insert l/r of 32 ? : l
Do you want to continue (y/n) : y

Enter the element : 80
Where to insert l/r of 60 ? : r
Where to insert l/r of 73 ? : r
Do you want to continue (y/n) : y

Enter the element : 68
Where to insert l/r of 60 ? : r
Where to insert l/r of 73 ? : l
Do you want to continue (y/n) : y

Enter the element : 41
Where to insert l/r of 60 ? : l
Where to insert l/r of 32 ? : r
Do you want to continue (y/n) : n

Binary tree has been created.
```

Display the elements in the tree and the number of elements

```
Enter your choice : 4
Displaying in inorder :
 12 32 41 60 68 73 80

Enter your choice : 5
Number of elements in the tree : 7
```

Deleting 3 items : 41, 73 and 12

```
Enter your choice : 3
Enter the value you wish to delete : 41
The element 41 has been deleted.

Enter your choice : 4
Displaying in inorder :
 12 32 60 68 73 80

Enter your choice : 3
Enter the value you wish to delete : 73
The element 73 has been deleted.

Enter your choice : 4
Displaying in inorder :
 12 32 60 68 80

Enter your choice : 3
Enter the value you wish to delete : 12
The element 12 has been deleted.

Enter your choice : 4
Displaying in inorder :
 32 60 68 80
```

Displaying the size after deletion

```
Enter your choice : 4
Displaying in inorder :
 32 60 68 80

Enter your choice : 5
Number of elements in the tree : 4
```