DSL All Practical Codes ~ By HK_OFFICIAL_

- **Practical 1**

To study python basic concepts.

- **Practical 2**

A1 In second year computer engineering class, group A student's play cricket, group B students play badminton and group C students play football. Write a Python program using functions to compute following: -

a) List of students who play both cricket and badminton

b) List of students who play either cricket or badminton but not both

c) Number of students who play neither cricket nor badminton

d) Number of students who play cricket and football but not badminton

Program :

```
# Function to accept the names of students who play a specific sport
def accept_args(student_list, sport_name):
  num_students = int(input(f"Enter the total number of students who play {sport_name}: "))
  for i in range(num_students):
    name = input(f"Enter the name of student {i+1} who plays {sport_name}: ")
    student_list.append(name)

# Function to display the list of students who play a specific sport
def show(student_list, sport_name):
  print(f"\nStudents who play {sport_name}:")
  for name in student_list:
    print(name, end=" ")
  print()

# Function to search for a student name in a list (returns 1 if found, else 0)
def search_set(student_list, student_name):
  return 1 if student_name in student_list else 0
```

```
# Function to find the intersection of two sets (common students in both lists)
def find_intersection_set(set1, set2, result_set):
    for student in set1:
        if search_set(set2, student):
            result_set.append(student)

# Function to find elements in set1 that are not in set2 (difference)
def find_different_set(set1, set2, result_set):
    for student in set1:
        if not search_set(set2, student):
            result_set.append(student)

# Function to find the union of two sets (all unique students from both lists)
def find_union_set(set1, set2, result_set):
    result_set.extend(set1)
    for student in set2:
        if not search_set(set1, student):
            result_set.append(student)

# Main program function to display a menu and perform set operations
def main():
    cricket_players = []
    badminton_players = []
    football_players = []

    while True:
        print("\nMenu:")
        print("\t1. Accept the information")
        print("\t2. List of students who play both cricket and badminton")
        print("\t3. List of students who play either cricket or badminton but not both")
        print("\t4. Number of students who play neither cricket nor badminton")
        print("\t5. Number of students who play cricket and football but not badminton")
        print("\t6. Exit")
```

```
choice = int(input("Enter your choice: "))
result = []

if choice == 6:
  print("Thank you!")
  break

elif choice == 1:
  accept_args(cricket_players, "cricket")
  accept_args(badminton_players, "badminton")
  accept_args(football_players, "football")
  show(cricket_players, "cricket")
  show(badminton_players, "badminton")
  show(football_players, "football")

elif choice == 2:
  find_intersection_set(cricket_players, badminton_players, result)
  show(result, "both cricket and badminton")

elif choice == 3:
  union_result = []
  intersection_result = []
  find_union_set(cricket_players, badminton_players, union_result)
  find_intersection_set(cricket_players, badminton_players, intersection_result)
  find_different_set(union_result, intersection_result, result)
  show(result, "either cricket or badminton but not both")

elif choice == 4:
  union_result = []
  find_union_set(cricket_players, badminton_players, union_result)
  find_different_set(football_players, union_result, result)
  show(result, "neither cricket nor badminton")
  print("Number of students who play neither cricket nor badminton: ",len(result))
```

```
    elif choice == 5:
      intersection_result = []
      find_intersection_set(cricket_players, football_players, intersection_result)
      find_different_set(intersection_result, badminton_players, result)
      show(result, "cricket and football but not badminton")
      print("Number of students who play cricket and football but not badminton: ",len(result))

    else:
      print("Invalid choice, please try again.")

# Run the main function to start the program
main()
```

- **Practical 3**

A2 Write a Python program to store marks scored in subject “Fundamental of Data Structure” by N students in the class. Write functions to compute following:

a) The average score of class

b) Highest score and lowest score of class

c) Count of students who were absent for the test

d) Display mark with highest frequency

Program :

```
def average(marks):
  total = 0
  count = 0
  for mark in marks:
    if mark != -999:
      total += mark
      count += 1
  if count > 0:
    avg = total / count
  else:
    avg = 0
  print("Average Marks:", avg)
```

```
def highest(marks):
  max_marks = -999
  for mark in marks:
    if mark != -999 and mark > max_marks:
      max_marks = mark
  print("Highest Marks:", max_marks)

def lowest(marks):
  min_marks = 999
  for mark in marks:
    if mark != -999 and mark < min_marks:
      min_marks = mark
  print("Lowest Marks:", min_marks)

def absent(marks):
  count = 0
  for mark in marks:
    if mark == -999:
      count += 1
  print("Absent Students:", count)

def most_frequent(marks):
  freq = {}
  for mark in marks:
    if mark != -999:
      freq[mark] = freq.get(mark, 0) + 1

  most_freq = None
  highest_count = 0
  for mark, count in freq.items():
    if count > highest_count:
      most_freq = mark
      highest_count = count
  print("Most Frequent Marks:", most_freq, "with Frequency:", highest_count)

marks = []
num_students = int(input("Enter total number of students: "))
for i in range(num_students):
  mark = int(input(f"Enter marks for student {i + 1}: "))
  marks.append(mark)

while True:
  print("\n1. Average Marks")
  print("2. Highest Marks")
  print("3. Lowest Marks")
  print("4. Absent Students")
  print("5. Most Frequent Marks")
  print("6. Exit")

  choice = int(input("Enter your choice: "))

  if choice == 1:
    average(marks)
  elif choice == 2:
    highest(marks)
  elif choice == 3:
    lowest(marks)
  elif choice == 4:
    absent(marks)
  elif choice == 5:
    most_frequent(marks)
  elif choice == 6:
    print("Thank you!")
    break
  else:
    print("Invalid choice! Please try again.")
```

- **Practical 4**

A-9 Write a Python program to compute following computation on matrix:

a)Addition of two matrices

b) Subtraction of two matrices

c) Multiplication of two matrices

d) Transpose of a matrix

Program :

```
def main():
  while True:
    print("Matrix Operations Menu:")
    print("1. accept Matrices")
    print("2. Display Matrices")
    print("3. Add Matrices")
    print("4. Subtract Matrices")
    print("5. Multiply Matrices")
    print("6. Transpose Matrices")
    print("7. Exit")

    choice = int(input("Enter your choice: "))

    if choice == 1:
      row1=int(input("Enter total number of rows  for matrix 1: "))
      col1=int(input("Enter total number of colummns for matrix 1: "))
      m1=[]

      print("fisrt matrix=")
      res=accept(row1,m1,col1)
      row2=int(input("Enter total number of rows for matrix 2 : "))
      col2=int(input("Enter total number of colummns for matrix 2: "))
      m2=[]
      print("second matrix=")
      res1=accept(row2,m2,col2)
```

```
        elif choice == 2:
          display(row1,res,col1)
          display(row2,res1,col2)

        elif choice == 3:
          add(row1,res,col1,res1,row2,col2)

        elif choice == 4:
          sub(row1,res,col1,res1,row2,col2)
        elif choice == 5:
          mul(row1,col1,row2,col2,res,res1 )

        elif choice == 6:
          transpose(res)

        elif choice == 7:
          break

        else:
          print("Invalid choice. Please enter a valid option.")

def accept(row,m,col):


  for i in range(0,row):
    x=[]
    for j in range(0,col):
      x.append(int(input("enter elements :\n")))

    m.append(x)

  return(m)
```

```
def display (row,res,col):
   print("\n")
   for i in range(0,row):


      for j in range(0,col):
         print(res[i][j],end="    ")


      print("\n")
def add(row1,res,col1,res1,row2,col2):
   if(row1==row2 and col1==col2):
      print("addition of two matries=")
      for i in range(0,row1):


         for j in range(0,col1):
            print(res[i][j]+res1[i][j],end="    ")
         print("\n")
   else:
      print ("number of col and row is difffrent so addition is not possible")
def sub(row1,res,col1,res1,row2,col2):
   if(row1==row2 and col1==col2):
      print("substraction of two matries=")
      for i in range(0,row1):


         for j in range(0,col1):
            print((res[i][j])-(res1[i][j]),end="    ")
         print("\n")
   else:
      print ("number of col and row is difffrent so substraction is not possible")
def mul(row1,col1,row2,col2,res1,res2):
   if (col1==row2):
      result=[]
      print("multiplication of matries")
      for i in range(0,row1):
         row = []
```

```
          for k in range(0,col2):
            element=0
            for j in range(0,col1):
              element += (res1[i][j] * res2[j][k])
            row.append(element)
          result.append(row)
        #return (result)
        for row in result:
          for element in row:
            print(element,end=" ")
          print()
      else:
        print("mul is not possible")
def transpose(matrix):
  tran=[]
  for i in range (len (matrix)):
    r=[]
    for j in range(len(matrix[0])):
      r.append(matrix[j][i])
    tran.append(r)
  print("transpose of matrix")
  for row in tran:
    for element in row:
      print(element,end=" ")
    print()

main()
```

- **Practical 5**

Write a python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using

a) Selection Sort

b) Bubble sort and display top five scores.

Program :

```
def accept_array(A):
  n = int(input("Enter the total no. of students: "))
  for i in range(n):
    x = float(input(f"Enter the first year percentage of student {i + 1}: "))
    A.append(x)
  print("Array accepted successfully\n\n")

def display_array(A):
  n = len(A)
  if n == 0:
    print("\nNo records in the database")
  else:
    print("Array of FE Marks:", end=' ')
    for i in range(n):
      print(f"{A[i]:.2f}", end=' ')
    print("\n")

def Selection_sort(A):
  n = len(A)
  for pos in range(n-1):
    min_ind = pos
    for i in range(pos + 1, n):
      if A[i] < A[min_ind]:
        min_ind = i
    A[pos], A[min_ind] = A[min_ind], A[pos]  # Swap in one line

def Bubble_sort(A):
  n = len(A)
  for pass_num in range(1, n):
    for i in range(n - pass_num):
      if A[i] < A[i + 1]:  # Sort in descending order
        A[i], A[i + 1] = A[i + 1], A[i]  # Swap in one line
```

```
def Main():
  A = []
  while True:
    print("\t1 : Accept & Display the FE Marks")
    print("\t2 : Selection Sort Ascending order")
    print("\t3 : Bubble sort Descending order and display top five scores")
    print("\t4 : Exit")
    ch = int(input("Enter your choice: "))
    if ch == 4:
      print("End of Program")
      break
    elif ch == 1:
      accept_array(A)
      display_array(A)
    elif ch == 2:
      print("Marks before sorting:")
      display_array(A)
      Selection_sort(A)
      print("Marks after sorting:")
      display_array(A)
    elif ch == 3:
      print("Marks before sorting:")
      display_array(A)
      Bubble_sort(A)
      print("Marks after sorting:")
      display_array(A)
      print("Top Five Scores:")
      top_scores = A[:5] if len(A) >= 5 else A
      for score in top_scores:
        print(f"\t{score:.2f}")
    else:
      print("Wrong choice entered! Try again.")

Main()
```

- **Practical 6**

Write a python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using

1.insertion sort

2.Shell Sort

Program :

```
import array as a
def insertion_sort(m,n):
  for i in range(1, n):
    key = m[i]
    j = i - 1

    while j >= 0 and key < m[j]:
      m[j + 1] = m[j]
      j -= 1

    m[j + 1] = key
  print("Marks of students after performing insertion Sort on the list : ")
  for i in range(len(m)):
    print("%.2f"%m[i])
def shell_sort(a,n):
  gap = n // 2
  while gap > 0:
    for i in range(gap, n):
      temp = a[i]
      j = i
      while j >= gap and a[j - gap] > temp:
        a[j] = a[j - gap]
        j -= gap
      a[j] = temp
    gap //= 2
    print("marks after shell sort :")
    for n in range (0,n):
        print("%.2f"%a[n])
def main():
    arr=a.array('f',[])
    l=int(input("enter number of student :"))
    print ("enetr marks of student")
    for i in range(0,l):
        print("student ",i+1)
        e=float(input())
        arr.append(e)
    print("student marks before sorting")
    for n in range (0,l):
        print("%.2f"%arr[n])
        flag=1;
    while True:
        print("Menu:")
        print("1. insertion Sort of the marks")
        print("2. shell Sort of the marks")
        print("3. top 5 student")
        print("4. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            insertion_sort(arr,l)

        elif choice == 2:
            shell_sort(arr,l)

        elif choice == 3:
            if (l<5):
                print (l ,"topper are :")
                for t in range (l-1,0-1,-1):
```

```
                print("%.2f" %arr[t])
            else:
                print("topper  students :")
                for k in range (l-1,l-6,-1):
                    print("%.2f" %arr[k])

        elif choice == 4:

            break

        else:
            print("Invalid choice. Please enter a valid option.")
main()
```

- **Practical 7**

B-16 Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using quick sort and display top five scores.

Program :

```
def partition(m, lb, ub):
    pivot = m[lb]
    s = lb + 1
    e = ub
    while s <= e:
        while s <= ub and m[s] <= pivot:
            s += 1
        while m[e] > pivot:
            e -= 1
        if s < e:
            m[s], m[e] = m[e], m[s]

    m[e], m[lb] = m[lb], m[e]
    return e

def quick_sort(m, lb, ub):
    if lb < ub:
        loc = partition(m, lb, ub)
        quick_sort(m, lb, loc - 1)
        quick_sort(m, loc + 1, ub)

def display(m):
    print("\nSorted student percentages:")
    for i, perc in enumerate(m, 1):
        print(f"Student {i}: {perc:.2f}%")

def main():
    l = int(input("Enter the number of students: "))
    arr = [float(input(f"Enter percentage for Student {i+1}: ")) for i in range(l)]

    print("\nStudents' percentages before sorting:")
    for i, perc in enumerate(arr, 1):
        print(f"Student {i}: {perc:.2f}%")

    while True:
        print("\nMenu:")
        print("1. Sort percentages")
        print("2. Show Top 5 students")
        print("3. Exit")

        choice = int(input("Enter your choice: "))

        if choice == 1:
            quick_sort(arr, 0, l - 1)
            display(arr)
        elif choice == 2:
            top = min(l, 5)
            print("\nTop students:")
            for i in range(top):
                print(f"Student {i+1}: {arr[l-i-1]:.2f}%")
        elif choice == 3:
            print("Exiting program.")
            break
        else:
            print("Invalid choice, please try again.")

if __name__ == "__main__":
    main()
```

- **Practical 8**

Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third and final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write C++ program to maintain club member‘s information using singly linked list. Store student PRN and Name. Write functions to:

a) Add and delete the members as well as president or even secretary.

b) Compute total number of members of club

c) Display members

d) Two linked lists exists for two divisions. Concatenate two lists.

Program :

```
#include <iostream>
#include <cstring>
using namespace std;

struct Node {
    int prn, rollno;
    char name[50];
    Node* next;
};
```

```
class Info {
    Node* head = NULL;
public:
    Node* create();
    void insertAtBegin();
    void insertAtEnd();
    void insertAfter(int rollno);
    void deleteAtBegin();
    void deleteAtEnd();
    void deleteByPrn(int prn);
    void display();
    void count();
    void concat(Info& other);
};

Node* Info::create() {
    Node* newNode = new Node;
    cout << "Enter name: ";
    cin >> newNode->name;
    cout << "Enter PRN: ";
    cin >> newNode->prn;
    cout << "Enter Roll No: ";
    cin >> newNode->rollno;
    newNode->next = NULL;
    return newNode;
}

void Info::insertAtBegin() {
    Node* newNode = create();
    newNode->next = head;
    head = newNode;
}
```

```
void Info::insertAtEnd() {
   Node* newNode = create();
   if (head == NULL) {
      head = newNode;
   } else {
      Node* temp = head;
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->next = newNode;
   }
}

void Info::insertAfter(int rollno) {
   Node* newNode = create();
   Node* temp = head;
   while (temp != NULL && temp->rollno != rollno) {
      temp = temp->next;
   }
   if (temp != NULL) {
      newNode->next = temp->next;
      temp->next = newNode;
   } else {
      cout << "Roll number not found.\n";
   }
}

void Info::deleteAtBegin() {
   if (head != NULL) {
      Node* temp = head;
      head = head->next;
      delete temp;
   } else {
      cout << "List is empty.\n";
```

```
    }
}

void Info::deleteAtEnd() {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }
    if (head->next == NULL) {
        delete head;
        head = NULL;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL && temp->next->next != NULL) {
        temp = temp->next;
    }
    delete temp->next;
    temp->next = NULL;
}

void Info::deleteByPrn(int prn) {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }
    if (head->prn == prn) {
        Node* temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Node* temp = head;
    while (temp->next != NULL && temp->next->prn != prn) {
        temp = temp->next;
    }
    if (temp->next != NULL) {
        Node* toDelete = temp->next;
        temp->next = temp->next->next;
        delete toDelete;
    } else {
        cout << "PRN not found.\n";
    }
}

void Info::display() {
    if (head == NULL) {
        cout << "List is empty.\n";
        return;
    }
    Node* temp = head;
    cout << "PRN\tRoll No\tName\n";
    while (temp != NULL) {
        cout << temp->prn << "\t" << temp->rollno << "\t" << temp->name << "\n";
        temp = temp->next;
    }
}

void Info::count() {
    int count = 0;
    Node* temp = head;
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }
    cout << "Number of members: " << count << "\n";
}
```

```
void Info::concat(Info& other) {
    if (head == NULL) {
        head = other.head;
    } else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = other.head;
    }
    other.head = NULL;
}

int main() {
    Info list1, list2;
    int choice, rollno, prn;
    char cont = 'y';

    while (cont == 'y' || cont == 'Y') {
        cout << "\nChoose an option:\n";
        cout << "1. Insert at beginning\n";
        cout << "2. Insert at end\n";
        cout << "3. Insert after a roll number\n";
        cout << "4. Delete from beginning\n";
        cout << "5. Delete from end\n";
        cout << "6. Delete by PRN\n";
        cout << "7. Count members\n";
        cout << "8. Display list\n";
        cout << "9. Concatenate two lists\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: list1.insertAtBegin(); break;
            case 2: list1.insertAtEnd(); break;
            case 3:
                cout << "Enter roll number: ";
                cin >> rollno;
                list1.insertAfter(rollno);
                break;
            case 4: list1.deleteAtBegin(); break;
            case 5: list1.deleteAtEnd(); break;
            case 6:
                cout << "Enter PRN: ";
                cin >> prn;
                list1.deleteByPrn(prn);
                break;
            case 7: list1.count(); break;
            case 8: list1.display(); break;
            case 9: list1.concat(list2); break;
            default: cout << "Invalid choice.\n"; break;
        }

        cout << "Do you want to continue? (y/n): ";
        cin >> cont;
    }

    return 0;
}
```

- **Practical 9**

Second year Computer Engineering class, set A of students like Vanilla Ice-cream and set B of students like butterscotch ice-cream.

Write C++ program to store two sets using linked list. compute and display-

a) Set of students who like both vanilla and butterscotch

b) Set of students who like either vanilla or butterscotch or not both

c) Number of students who like neither vanilla nor butterscotch

Program :

```
#include <iostream>
using namespace std;

struct Node {
  int roll;
  Node* next;
};

class Info {
  Node *headVanila = nullptr, *headButterscotch = nullptr, *headAll = nullptr;

public:
  void addStudent(int roll, Node*& head);
  void displayList(Node* head);
  void allStudents();
  void vanila();
  void butterscotch();
  void bothLikes();
  void onlyVanila();
  void onlyButterscotch();
  void neitherLikes();
};

void Info::addStudent(int roll, Node*& head) {
  Node* newNode = new Node;
  newNode->roll = roll;
  newNode->next = nullptr;

  if (!head) {
    head = newNode;
  } else {
    Node* temp = head;
    while (temp->next) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

void Info::displayList(Node* head) {
    Node* temp = head;
    while (temp) {
        cout << temp->roll << " ";
        temp = temp->next;
    }
    cout << endl;
}

void Info::allStudents() {
    int n, roll;
    cout << "Enter number of students: ";
    cin >> n;

    cout << "Enter roll numbers of all students:\n";
    for (int i = 0; i < n; ++i) {
        cin >> roll;
        addStudent(roll, headAll);
    }

    cout << "All students: ";
    displayList(headAll);
}

void Info::vanila() {
    int n, roll;
    cout << "Enter number of students who like vanilla: ";
    cin >> n;
    cout << "Enter roll numbers of students who like vanilla:\n";
    for (int i = 0; i < n; ++i) {
        cin >> roll;
        addStudent(roll, headVanila);
    }

    cout << "Students who like vanilla: ";
    displayList(headVanila);
}

void Info::butterscotch() {
    int n, roll;
    cout << "Enter number of students who like butterscotch: ";
    cin >> n;

    cout << "Enter roll numbers of students who like butterscotch:\n";
    for (int i = 0; i < n; ++i) {
        cin >> roll;
        addStudent(roll, headButterscotch);
    }

    cout << "Students who like butterscotch: ";
    displayList(headButterscotch);
}

void Info::bothLikes() {
    cout << "Students who like both vanilla and butterscotch: ";
    Node* tempVanila = headVanila;
    while (tempVanila) {
        Node* tempButterscotch = headButterscotch;
        while (tempButterscotch) {
            if (tempVanila->roll == tempButterscotch->roll) {
                cout << tempVanila->roll << " ";
            }
            tempButterscotch = tempButterscotch->next;
        }
        tempVanila = tempVanila->next;
    }
    cout << endl;
}

void Info::onlyVanila() {
    cout << "Students who like only vanilla: ";
    Node* tempVanila = headVanila;
    while (tempVanila) {
        Node* tempButterscotch = headButterscotch;
        bool found = false;
        while (tempButterscotch) {
            if (tempVanila->roll == tempButterscotch->roll) {
                found = true;
                break;
            }
            tempButterscotch = tempButterscotch->next;
        }
        if (!found) {
            cout << tempVanila->roll << " ";
        }
        tempVanila = tempVanila->next;
    }
    cout << endl;
}

void Info::onlyButterscotch() {
    cout << "Students who like only butterscotch: ";
    Node* tempButterscotch = headButterscotch;
    while (tempButterscotch) {
        Node* tempVanila = headVanila;
        bool found = false;
        while (tempVanila) {
            if (tempButterscotch->roll == tempVanila->roll) {
                found = true;
                break;
            }
            tempVanila = tempVanila->next;
        }
        if (!found) {
            cout << tempButterscotch->roll << " ";
        }
        tempButterscotch = tempButterscotch->next;
    }
    cout << endl;
}

void Info::neitherLikes() {
    cout << "Students who like neither vanilla nor butterscotch: ";
    Node* tempAll = headAll;
    while (tempAll) {
        bool found = false;
        Node* tempVanila = headVanila;
        while (tempVanila) {
            if (tempAll->roll == tempVanila->roll) {
                found = true;
                break;
            }
            tempVanila = tempVanila->next;
        }

        Node* tempButterscotch = headButterscotch;
        while (tempButterscotch) {
            if (tempAll->roll == tempButterscotch->roll) {
                found = true;
            break;
          }
          tempButterscotch = tempButterscotch->next;
        }

        if (!found) {
          cout << tempAll->roll << " ";
        }
        tempAll = tempAll->next;
      }
      cout << endl;
    }

int main() {
  Info info;
  int choice;
  char cont;

  do {
    cout << "\n1. Enter all students' roll numbers\n";
    cout << "2. Enter students who like vanilla\n";
    cout << "3. Enter students who like butterscotch\n";
    cout << "4. Display students who like both vanilla and butterscotch\n";
    cout << "5. Display students who like only vanilla\n";
    cout << "6. Display students who like only butterscotch\n";
    cout << "7. Display students who like neither vanilla nor butterscotch\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
      case 1: info.allStudents(); break;
      case 2: info.vanila(); break;
      case 3: info.butterscotch(); break;
      case 4: info.bothLikes(); break;
```

```
            case 5: info.onlyVanila(); break;
            case 6: info.onlyButterscotch(); break;
            case 7: info.neitherLikes(); break;
            default: cout << "Invalid choice!" << endl;
        }

        cout << "Do you want to continue (y/n)? ";
        cin >> cont;
    } while (cont == 'y' || cont == 'Y');

    return 0;
}
```

- **Practical 10**

In any language program mostly syntax error occurs due to unbalancing delimiter such as (),{},[].

Write C++ program using stack to check

whether given expression is well parenthesized or not.

Program :

```
#include <iostream>
#include <string.h>
using namespace std;

class Braces {
    char st[20];
    int top;

public:
    void push(char a);
    void pop();
    void checkExpression();
};
```

```
void Braces::push(char a) {
  top++;
  st[top] = a;
}

void Braces::pop() {
  top--;
}

void Braces::checkExpression() {
  char ch[20];
  int i = 0;
  top = -1;
  cout << "Enter the expression: ";
  cin >> ch;

  while (ch[i] != '\0') {
    if (ch[i] == '{' || ch[i] == '[' || ch[i] == '(') {
      push(ch[i]);
    } else if (ch[i] == '}') {
      if (top != -1 && st[top] == '{') {
        pop();
      } else {
        cout << "\nMatching opening brace '{' is not found!";
        return;
      }
    } else if (ch[i] == ']') {
      if (top != -1 && st[top] == '[') {
        pop();
      } else {
        cout << "\nMatching opening brace '[' is not found!";
        return;
      }
    } else if (ch[i] == ')') {
            if (top != -1 && st[top] == '(') {
                pop();
            } else {
                cout << "\nMatching opening brace '(' is not found!";
                return;
            }
        }
        i++;
    }

    if (top == -1) {
        cout << "\nExpression is well parenthesized.\n";
    } else {
        cout << "\nExpression is not well parenthesized.\n";
    }
}

int main() {
    Braces b;
    int choice;

    do {
        cout << "\nMenu:";
        cout << "\n1. Check if the expression is well parenthesized";
        cout << "\n2. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                b.checkExpression();
                break;
            case 2:
                cout << "Exiting program...\n";
            break;
        default:
            cout << "Invalid choice! Please try again.\n";
    }

  } while (choice != 2);

  return 0;
}
```

- **Practical 11**

Implement C++ program for expression conversion as infix to postfix and its evaluation using stack based on given conditions:

1. Operands and operator, both must be single character.
2. Input Postfix expression must be in a desired format.
3. Only '+', '-', '*' and '/ ' operators are expected.

Program :

```
#include <iostream>
using namespace std;

class Stack {
public:
  char stack_array[50];
  int top;

  Stack() {
    top = -1;
  }

  void push(char symbol) {
    if (top < 49) {
      stack_array[++top] = symbol;
    } else {
      cout << "\nStack overflow!\n";
    }
  }

  char pop() {
    if (top >= 0) {
      return stack_array[top--];
    } else {
      return '#'; // Indicating stack is empty
    }
  }

  bool empty() {
    return top == -1;
  }

  bool full() {
    return top == 49;
  }

  int precedence(char symbol) {
    switch (symbol) {
      case '+': case '-': return 1;
      case '*': case '/': return 2;
      case '(': return 0;
      default: return -1;
    }
  }

  void convertToPostfix() {
    char infix[50], postfix[50], entry;
    int p = 0;
    cout << "\nEnter an infix expression: ";
    cin >> infix;
```

```
        for (int i = 0; infix[i] != '\0'; i++) {
            if (infix[i] == '(') {
                push(infix[i]);
            } else if (infix[i] == ')') {
                while ((entry = pop()) != '(') {
                    postfix[p++] = entry;
                }
            } else if (infix[i] == '+' || infix[i] == '-' || infix[i] == '*' || infix[i] == '/') {
                while (!empty() && precedence(infix[i]) <= precedence(stack_array[top])) {
                    postfix[p++] = pop();
                }
                push(infix[i]);
            } else {
                postfix[p++] = infix[i];
            }
        }

        while (!empty()) {
            postfix[p++] = pop();
        }

        postfix[p] = '\0'; // Null-terminate the postfix expression
        cout << "\nPostfix expression: " << postfix << endl;
    }
};

int main() {
    Stack expr;
    char choice = 'y';
    while (choice == 'y') {
        expr.convertToPostfix();
        cout << "\nDo you want to continue? (y/n): ";
        cin >> choice;
    }
    return 0;
}
```

- **Practical 12**

Queues are frequently used in computer programming, and a typical example is the creation of a job queue by an operating system. If the operating system does not use priorities, then the jobs are processed in the order they enter the system. Write C++ program for simulating job queue.

Write functions to add job and delete job from queue.

Program :

```
#include <iostream>
#define MAX 10
using namespace std;

struct Queue {
    int data[MAX];
    int front, rear;
};

class QueueOperations {
private:
    Queue q;

public:
    QueueOperations() {
        q.front = q.rear = -1;
    }

    bool isEmpty() {
        return q.front == q.rear;
    }

    bool isFull() {
        return q.rear == MAX - 1;
    }

    void enqueue(int x) {
        if (!isFull()) {
            q.data[++q.rear] = x;
            cout << "Job inserted: " << x << endl;
        } else {
            cout << "Queue Overflow! Cannot insert job." << endl;
        }
    }

    void dequeue() {
        if (!isEmpty()) {
            cout << "Deleted Job: " << q.data[++q.front] << endl;
        } else {
            cout << "Queue Underflow! No job to delete." << endl;
        }
    }

    void display() {
        if (!isEmpty()) {
            cout << "Jobs in Queue: ";
            for (int i = q.front + 1; i <= q.rear; i++) {
                cout << q.data[i] << " ";
            }
            cout << endl;
        } else {
            cout << "Queue is empty!" << endl;
        }
    }
};

int main() {
    QueueOperations queue;
    int choice, job;

    do {
      cout << "\nMenu:\n"
          << "1. Insert Job\n"
          << "2. Delete Job\n"
          << "3. Display Jobs\n"
          << "4. Exit\n"
          << "Enter your choice: ";
      cin >> choice;

      switch (choice) {
        case 1:
          cout << "Enter job data: ";
          cin >> job;
          queue.enqueue(job);
          break;
        case 2:
          queue.dequeue();
          break;
        case 3:
          queue.display();
          break;
        case 4:
          cout << "Exiting Program..." << endl;
          break;
        default:
          cout << "Invalid choice! Please try again." << endl;
      }
    } while (choice != 4);

    return 0;
  }
```

- **Practical 13**

A double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Obtain a data representation mapping a deque into a one-dimensional array. Write C++ program to simulate deque with functions to add and delete elements from either end of the deque.

Program :

```
#include <iostream>
#define MAX 10
using namespace std;

struct Queue {
  int arr[MAX];
  int front, rear;
};

void init(Queue &q) {
  q.front = q.rear = -1;
}

bool isEmpty(const Queue &q) {
  return q.front == -1;
}

bool isFull(const Queue &q) {
  return (q.rear + 1) % MAX == q.front;
}

void insertFront(Queue &q, int data) {
  if (isFull(q)) {
    cout << "Queue is full! Cannot insert at front.\n";
  } else {
    if (isEmpty(q)) {
      q.front = q.rear = 0;
    } else {
      q.front = (q.front - 1 + MAX) % MAX;
    }
    q.arr[q.front] = data;
    cout << "Data inserted at front: " << data << endl;
  }
}

void insertRear(Queue &q, int data) {
  if (isFull(q)) {
    cout << "Queue is full! Cannot insert at rear.\n";
  } else {
    if (isEmpty(q)) {
      q.front = q.rear = 0;
    } else {
      q.rear = (q.rear + 1) % MAX;
    }
    q.arr[q.rear] = data;
    cout << "Data inserted at rear: " << data << endl;
  }
}

int deleteFront(Queue &q) {
  if (isEmpty(q)) {
    cout << "Queue is empty! Cannot delete from front.\n";
    return -1;
  } else {
    int data = q.arr[q.front];
    if (q.front == q.rear) {
      init(q); // Reset the queue when it's empty
    } else {
      q.front = (q.front + 1) % MAX;
    }
```

```
        return data;
    }
}

int deleteRear(Queue &q) {
    if (isEmpty(q)) {
        cout << "Queue is empty! Cannot delete from rear.\n";
        return -1;
    } else {
        int data = q.arr[q.rear];
        if (q.front == q.rear) {
            init(q); // Reset the queue when it's empty
        } else {
            q.rear = (q.rear - 1 + MAX) % MAX;
        }
        return data;
    }
}

void display(const Queue &q) {
    if (isEmpty(q)) {
        cout << "Queue is empty!\n";
    } else {
        cout << "Queue elements: ";
        int i = q.front;
        while (i != q.rear) {
            cout << q.arr[i] << " ";
            i = (i + 1) % MAX;
        }
        cout << q.arr[q.rear] << endl;
    }
}

int main() {
```

```
Queue q;
int choice, data;
init(q);

do {
  cout << "\n1. Insert at Front";
  cout << "\n2. Insert at Rear";
  cout << "\n3. Delete from Front";
  cout << "\n4. Delete from Rear";
  cout << "\n5. Display Queue";
  cout << "\n6. Exit";
  cout << "\nEnter your choice: ";
  cin >> choice;

  switch (choice) {
    case 1:
      cout << "Enter data to insert at front: ";
      cin >> data;
      insertFront(q, data);
      break;
    case 2:
      cout << "Enter data to insert at rear: ";
      cin >> data;
      insertRear(q, data);
      break;
    case 3:
      data = deleteFront(q);
      if (data != -1) {
        cout << "Deleted data from front: " << data << endl;
      }
      break;
    case 4:
      data = deleteRear(q);
      if (data != -1) {
            cout << "Deleted data from rear: " << data << endl;
          }
          break;
        case 5:
          display(q);
          break;
        case 6:
          cout << "Exiting program...\n";
          break;
        default:
          cout << "Invalid choice. Please try again.\n";
      }
  } while (choice != 6);

  return 0;
}
```

- **Practical 14**

Pizza parlor accepting maximum M orders. Orders are served in first come first served basis. Order once placed cannot be cancelled. Write C++ program to simulate the system using circular queue using array.

Program :

```
#include <iostream>
using namespace std;

#define SIZE 5

class PizzaParlor {
  int porder[SIZE];
  int front, rear;

public:
  PizzaParlor() {
      front = rear = -1;
  }

  bool isFull() {
      return (front == 0 && rear == SIZE - 1) || (front == rear + 1) % SIZE;
  }

  bool isEmpty() {
      return front == -1;
  }

  void acceptOrder(int item);
  void makePayment(int n);
  void viewPendingOrders();
};

void PizzaParlor::acceptOrder(int item) {
  if (isFull()) {
      cout << "\nSorry, we cannot accept more orders at the moment.\n";
  } else {
      if (front == -1) {
          front = rear = 0;
      } else {
          rear = (rear + 1) % SIZE;
      }
      porder[rear] = item;
      cout << "Order accepted successfully.\n";
  }
}

void PizzaParlor::makePayment(int n) {
  if (isEmpty()) {
      cout << "\nNo pending orders to process.\n";
  } else {
      cout << "\nDelivering orders: ";
      for (int i = 0; i < n; i++) {
          int item = porder[front];
          cout << "\t" << item;
          if (front == rear) {
              front = rear = -1;  // Reset if the queue is empty
          } else {
              front = (front + 1) % SIZE;
          }
      }
      cout << "\nTotal amount: " << n * 100 << " units.\n";
      cout << "Thank you for visiting. See you again!\n";
  }
}

void PizzaParlor::viewPendingOrders() {
  if (isEmpty()) {
      cout << "\nNo pending orders.\n";
  } else {
      cout << "\nPending orders are: ";
      int temp = front;
      while (temp != rear) {
          cout << "\t" << porder[temp];
          temp = (temp + 1) % SIZE;
      }
      cout << "\t" << porder[rear] << endl;
  }
}

int main() {
  PizzaParlor p1;
  int choice, pizzaType, quantity;


  do {
```

```
    cout << "\n***** Welcome to Pizza Parlor *****";
    cout << "\n1. Accept Order";
    cout << "\n2. Make Payment";
    cout << "\n3. View Pending Orders";
    cout << "\n4. Exit";
    cout << "\nEnter your choice: ";
    cin >> choice;

    switch (choice) {
      case 1:
        cout << "\nWhat type of pizza would you like to order?\n";
        cout << "1. Veg Soya Pizza\n2. Veg Butter Pizza\n3. Egg Pizza\n";
        cout << "Enter your choice: ";
        cin >> pizzaType;
        p1.acceptOrder(pizzaType);
        break;
      case 2:
        cout << "\nHow many pizzas would you like to pay for? ";
        cin >> quantity;
        p1.makePayment(quantity);
        break;
      case 3:
        p1.viewPendingOrders();
        break;
      case 4:
        cout << "Exiting... Thank you for visiting!\n";
        break;
      default:
        cout << "Invalid choice, please try again.\n";
    }
  } while (choice != 4);

  return 0;
}
```