DAY 3

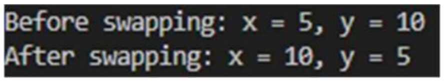
Student Name: Daksh **UID:** 22BCS15372
Branch: BE-CSE **Section/Group:** 620 - B
Date of Performance: 23/12/24

Problem 1

1. **Aim: Implement the function that swipe to variable using pass by reference**
2. **Code:**

```
#include <iostream> void
swap(int &a, int &b) { int
temp = a; a = b; b = temp;
} int main()
{ int x = 5;
int y = 10;
    cout << "Before swapping: x = "<<x << ", y = " << y << endl; swap(x,
    y);
    cout << "After swapping: x = "<<x<<", y = "<<y<<endl;
    return 0;
}
```

3. **Output:**



```
Before swapping: x = 5, y = 10
After swapping: x = 10, y = 5
```

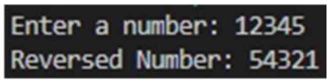
Problem 2

1. **Aim: Writer recursive function to compute the GCD of 2 numbers**
2. **Code:**

```
int gcd(int a, int b) {
    if (b == 0) {
    return a;
```

```
         }
         return gcd(b, a % b);
    } int main()
    { int num1,
    num2; cout
    << "Enter
    two
    integers: ";
    cin >>
    num1 >>
    num2; int
    result =
    gcd(num1,
    num2);
         cout << "GCD of "<<num1<<" and "<<num2<<" is: "<<result<<endl;
         return 0;
    }
```

3. Output:



```
Enter a number: 12345
Reversed Number: 54321
```

Problem 3

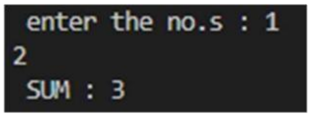
1. Aim: SUM OF TWO NO. USING FUNCTION

2. Code:

```
int sum(int x , int y)

{ return x+y;

}         int

main()
```

```
{ int x, y; cout<<"enter the no.s
   :          ";          cin>>x>>y;
   cout<<"SUM : "<<sum(x,y);
   return 0;
}
```

3. Output:



```
 enter the no.s : 1
2
 SUM : 3
```

Problem 4

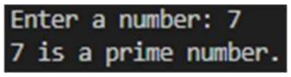
1. **Aim: Check if a Number is Prime**

2. **Problem Statement:** Check if a given number n is a prime number. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.
   To determine if a number is prime, iterate from 2 to √n and check if n is divisible by any number in this range. If it is divisible, it is not a prime number; otherwise, it is a prime
3. **Task:** Given an integer n, print "Prime" if the number is prime, or "Not Prime" if it is not
4. **Code:**

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

bool isPrime(int number) {
   if   (number   <=   1)   {
   return false;
```

```
        }
        for (int i = 2; i * i <= number; i++) {
            if (number % i == 0) { return
            false;
            }
        }
        return true;
    } int main()
    { int num;
        cout << "Enter a number: "; cin
        >> num;

        if (isPrime(num)) { cout << num << " is a prime
            number." << endl;
        } else { cout << num << " is not a prime number." <<
        endl; }
return 0;
    }
```

5. }Output:



```
Enter a number: 7
7 is a prime number.
```

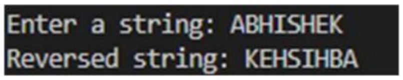
Problem 5

1. Aim: Write a function to reverse the string

2. Code:

```
string reverseString(const std::string& str) {
    string reversedStr = str; int n =
    reversedStr.length(); for (int i = 0; i < n / 2;
    ++i) { swap(reversedStr[i], reversedStr[n - i -
    1]);
    }
    return reversedStr;
} int main() { string input; cout <<
"Enter a string: "; getline(std::cin,
input);    string    output    =
reverseString(input);
    cout << "Reversed string: " << output << std::endl; return
    0;
}
```

3. Output:



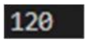
```
Enter a string: ABHISHEK
Reversed string: KEHSIHBA
```

Problem 6

1. **Aim: Fibonacci Series Using Recursion**
2. **Problem Statement:** The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1.
3. **Code:**

```
int fact(int n)
{ if (n == 0 || n == 1) {
    return 1;
    }
    return n* fact(n-1);
}        int
main()
{ int n = 5, ans;
    cout<<fact(n)
    ; return 0;
}
```

4. Output:



```
120
```

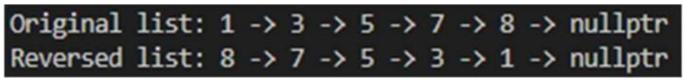
Problem 7

1. Aim: Reverse the LinkedList and return the reversed list

2. Code:

```
string reverseString(const std::string& str) { string
   reversedStr = str;

   int n = reversedStr.length(); for (int i = 0; i <
   n / 2; ++i) { swap(reversedStr[i],
   reversedStr[n - i - 1]);
   }
   return reversedStr;
} int main() { string input; cout <<
"Enter a string: "; getline(std::cin,
input);      string      output      =
reverseString(input);
   cout << "Reversed string: " << output << std::endl; return
   0;
}
```

3. Output:



```
Original list: 1 -> 3 -> 5 -> 7 -> 8 -> nullptr
Reversed list: 8 -> 7 -> 5 -> 3 -> 1 -> nullptr
```

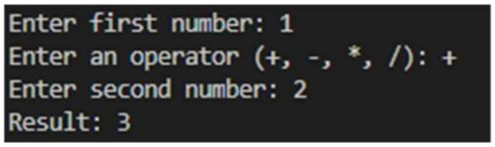
Problem 8

1. **Aim: write a c++ program to create a simple calculator that perform basic athematic operations like add, multiply, divide, sub**
2. **Code:**

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

void calculator(double num1, double num2, char operation) {
    switch (operation) { case '+':
            cout << "Result: " << num1 + num2 << endl;
        break; case '-':
            cout << "Result: " << num1 - num2 << endl; break;
        case '*':
            cout << "Result: " << num1 * num2 << endl; break;
        case '/':
            if (num2 != 0)
                cout << "Result: " << num1 / num2 << endl;
            else cout << "Error: Division by zero is not allowed." <<
                endl;
            break;
        default: cout << "Invalid operation. Please use +, -, *, or /." <<
            endl;
    }
}

int main() {
    double num1, num2; char
    operation;

    cout << "Enter first number: ";
    cin >> num1;
    cout << "Enter an operator (+, -, *, /): ";
```

```
          cin >> operation;
          cout << "Enter second number: "; cin
          >> num2;
calculator(num1, num2, operation);
return 0;
     }
```

3. Output:



```
Enter first number: 1
Enter an operator (+, -, *, /): +
Enter second number: 2
Result: 3
```

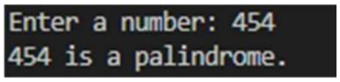
Problem 9

1. **Aim: write a c++ program check if the no. is palindrome or not using function.**
2. **Code:**

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


bool isPalindrome(int num) {
  int original = num; int
  reversed = 0;

  while (num > 0) { int digit = num % 10; //
    Extract the last digit
    reversed = reversed * 10 + digit; // Build the reversed
  number num /= 10; // Remove the last digit }

  return original == reversed; // Check if the original and reversed numbers
are equal
}

int main() { int
  number;
```

```
        cout << "Enter a number: "; cin
        >> number;

        if (isPalindrome(number)) { cout << number << "
          is a palindrome." << endl;
        } else { cout << number << " is not a palindrome." <<
        endl; }
        return 0;
      }
```

3. Output:



```
Enter a number: 454
454 is a palindrome.
```

Problem 10

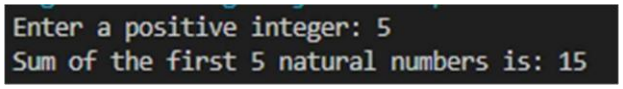
1. Aim: SUM OF NATURAL NO. USING RECURSION

2. Code:

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


      int sumOfNaturalNumbers(int n) { return
        (n * (n + 1)) / 2;
      }

      int main() { int
        n;
        cout << "Enter a positive integer: "; cin
        >> n;
if (n > 0) {
                cout << "Sum of the first " << n << " natural numbers is: " <<
      sumOfNaturalNumbers(n) << endl;
```

```
    } else { cout << "Please enter a positive integer." <<
    endl; }

    return 0;
}
```

3. **Output:**



```
Enter a positive integer: 5
Sum of the first 5 natural numbers is: 15
```

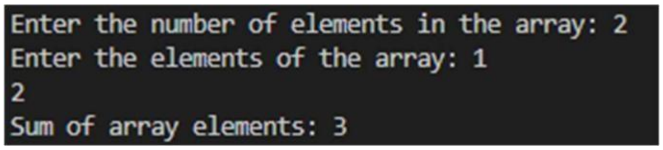
Problem 11

1. **Aim: SUM OF ARRAY ELEMENT USINNG RECURSION**
2. **Code:**

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


int sumOfArray(int arr[], int n) { if
    (n == 0) return 0;
    return arr[n - 1] + sumOfArray(arr, n - 1); }

int main() { int
    n;
    cout << "Enter the number of elements in the array: "; cin
    >> n;

    int arr[n];
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; ++i) { cin
        >> arr[i];
    }
```

```
          int sum = sumOfArray(arr, n);
          cout << "Sum of array elements: " << sum << endl;
return 0;
        }
```

3. Output:



```
Enter the number of elements in the array: 2
Enter the elements of the array: 1
2
Sum of array elements: 3
```

Problem 12

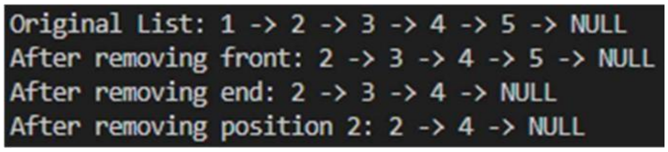
1. Aim: REMOVE LINKED LIST ELEMENT

2. Code:

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

struct Node { int
   data;
   Node* next;
   Node(int value) : data(value), next(nullptr) {}
};



void displayList(Node* head) {
   while (head) {
      cout << head->data << " -> "; head
      = head->next;
   }
   cout << "NULL" << endl;
}
Node* removeFront(Node* head) {
   if (!head) return nullptr; Node*
```

```
    temp = head; head = head->next;
    delete temp; return head;
}



Node* removeEnd(Node* head) {
    if (!head) return nullptr; if
    (!head->next) { delete head;
    return nullptr;
    }
    Node* temp = head;
    while (temp->next && temp->next->next) {
        temp = temp->next;
    } delete temp->next;
    temp->next = nullptr;
    return head;
}



Node* removeAtPosition(Node* head, int position) {
    if (position <= 0 || !head) return head; if (position
    == 1) return removeFront(head); Node* temp =
    head;
    for (int i = 1; i < position - 1 && temp->next; ++i) {
        temp = temp->next;
    }
    if (temp->next) {
        Node* toDelete = temp->next; temp->next
        = temp->next->next; delete toDelete;
    }      return
    head;
}

int main() {
    // Creating a linked list: 1 -> 2 -> 3 -> 4 -> 5
    Node* head = new Node(1); head->next = new
    Node(2); head->next->next = new Node(3);
    head->next->next->next = new Node(4); head-
    >next->next->next->next = new Node(5);

    cout << "Original List: "; displayList(head);

    // Removing the front element
    head = removeFront(head); cout
    << "After removing front: ";
    displayList(head);

    // Removing the last element
    head = removeEnd(head); cout
    << "After removing end: ";
    displayList(head);

    // Removing element at position 2
    head = removeAtPosition(head, 2);
    cout << "After removing position 2: ";
    displayList(head);

    return 0;
}
```

3. Output:



```
Original List: 1 -> 2 -> 3 -> 4 -> 5 -> NULL
After removing front: 2 -> 3 -> 4 -> 5 -> NULL
After removing end: 2 -> 3 -> 4 -> NULL
After removing position 2: 2 -> 4 -> NULL
```

Problem 13

1. **Aim:** Palindrome Linked List. Given The Head Of Simple Linked List. True If It Has A Palindrome.

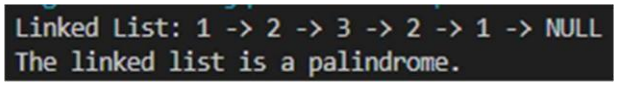
2. Code:

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

struct Node { int
   data;
   Node* next;
   Node(int value) : data(value), next(nullptr) {} };

// Function to add a new node to the end of the linked list
void append(Node*& head, int value) {
   if (!head) { head = new Node(value);
      return;
   }
   Node* temp = head; while
   (temp->next) {
      temp = temp->next;
   }
   temp->next = new Node(value);
}

// Function to check if the linked list is a palindrome
bool isPalindrome(Node* head) { if (!head || !head->next) return true; //
   Empty or single-element list is a
palindrome

   Node* slow = head;
   Node*  fast  =  head;
   stack<int> s;
```

```
         // Push the first half of the list onto the stack while
         (fast && fast->next) {
            s.push(slow->data);
            slow = slow->next; fast
            = fast->next->next;
         }

         // If the list has an odd number of elements, skip the middle element if
         (fast) slow = slow->next;

         // Compare the second half of the list with the stack
         while (slow) { if (slow->data != s.top()) return
         false; s.pop();
            slow = slow->next;
         }
return true;
     }

     // Function to display the linked list
     void displayList(Node* head) {
     while (head) { cout << head->data
     << " -> ";
            head = head->next;
         }
         cout << "NULL" << endl;
     }

     int main() {
         Node* head = nullptr;

         // Create a linked list: 1 -> 2 -> 3 -> 2 -> 1
         append(head, 1);
         append(head, 2);
         append(head, 3);
```

```
          append(head, 2);
          append(head, 1);

          cout << "Linked List: "; displayList(head);

          if (isPalindrome(head)) {
              cout << "The linked list is a palindrome." << endl;
          } else { cout << "The linked list is not a palindrome." <<
          endl; }
return 0;
          }
```

3. Output:



```
Linked List: 1 -> 2 -> 3 -> 2 -> 1 -> NULL
The linked list is a palindrome.
```

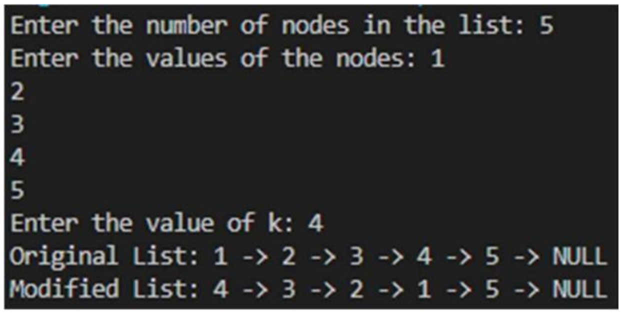
Problem 14

1. **Aim: Writer recursive function to compute the GCD of 2 numbers**
2. **Code:**

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

struct Node { int
    data;
    Node* next;
    Node(int value) : data(value), next(nullptr) {} };

// Function to reverse the first k nodes of the linked list
Node* reverseKGroup(Node* head, int k) { if
    (!head || k <= 1) return head;
    Node* prev = nullptr;
    Node*  curr  =  head;
    Node* next = nullptr;
    int count = 0;

    // Check if there are at least k nodes in the list
    Node* temp = head;
    for (int i = 0; i < k; ++i) {
        if (!temp) return head; // Not enough nodes to reverse
        temp = temp->next;
    }

    // Reverse the first k nodes while
    (curr && count < k) {
        next = curr->next;
        curr->next = prev;
        prev = curr; curr =
        next;
        count++;
    }

    // Recursively reverse the remaining nodes if
    (next) {
        head->next = reverseKGroup(next, k);
    }

    // Return the new head of the reversed list return
    prev;
}

// Function to append a node to the end of the list void
append(Node*& head, int value) {
    if  (!head)  {  head  =  new
        Node(value);
        return;
    }
    Node* temp = head; while
    (temp->next) {
        temp = temp->next;
    }
    temp->next = new Node(value);
}
// Function to display the linked list
void   displayList(Node*   head)   {
while (head) {
        cout << head->data << " -> ";
        head = head->next;
    }
    cout << "NULL" << endl;
}

int main() { Node* head
    =   nullptr;   int   n,   k,
    value;

    cout << "Enter the number of nodes in the list: "; cin
    >> n;

    cout << "Enter the values of the nodes: ";
    for (int i = 0; i < n; ++i) {
        cin        >>        value;
        append(head, value);
    }

    cout << "Enter the value of k: "; cin
    >> k;

    cout << "Original List: "; displayList(head);

    head = reverseKGroup(head, k);
```

```
                    cout << "Modified List: "; displayList(head);
return 0;
                }
```

3. Output:



```
Enter the number of nodes in the list: 5
Enter the values of the nodes: 1
2
3
4
5
Enter the value of k: 4
Original List: 1 -> 2 -> 3 -> 4 -> 5 -> NULL
Modified List: 4 -> 3 -> 2 -> 1 -> 5 -> NULL
```

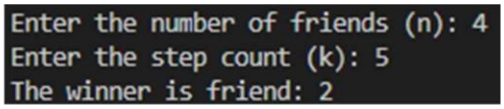
Problem 15

1. **Aim: FIND THE WINNER OF CIRCULAR GAME.**
2. **Code:**

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

        int findWinner(int n, int k) { vector<int> friends; for (int i
          = 1; i <= n; i++) { friends.push_back(i); // Initialize the
          circle of friends }
int index = 0; // Start at the first friend
          while (friends.size() > 1) { index = (index + k - 1) % friends.size(); //
             Find the index of the friend
        to remove friends.erase(friends.begin() + index); // Remove the friend
             from the
        circle
          }
```

```
    return friends[0]; // The last remaining friend is the winner }

int main() { int
    n, k;
    cout << "Enter the number of friends (n): "; cin
    >> n;
    cout << "Enter the step count (k): "; cin
    >> k;

    int winner = findWinner(n, k);
    cout << "The winner is friend: " << winner << endl;

    return 0;

}
```

3. Output:



```
Enter the number of friends (n): 4
Enter the step count (k): 5
The winner is friend: 2
```