NAME: Akshay raj singh

ROLL NO.:-

BT22CSH054

CODE:-

```
#include
<stdio.h>
#include
<stdlib.h>
#include
<math.h>

struct Node {
   float
   coefficient; int
   exponent;
   struct Node*
   next;
};

typedef struct Node Node;

Node* createNode(float coefficient, int
   exponent) { Node* newNode =
   (Node*)malloc(sizeof(Node)); if (newNode
   == NULL) {
      printf("Memory allocation
      failed.\n"); exit(1);
```

```
    }
    newNode->coefficient =
    coefficient; newNode-
    >exponent = exponent;
    newNode->next = NULL;
    return newNode;
}

void insertTerm(Node** header, float coefficient, int exponent) {
```

```
    Node* newNode = createNode(coefficient,
    exponent); if (*header == NULL) {
        *header = newNode;
        (*header)->next =
        *header;
    } else {
        Node* temp = *header;
        while (temp->next !=
            *header) { temp = temp-
            >next;
        }
        temp->next =
        newNode; newNode-
        >next = *header;
    }
}

Node* Pread() {
    Node* header =
    NULL; int
    numTerms;
    printf("Enter the number of terms in the polynomial: ");
    scanf("%d", &numTerms);

    for (int i = 0; i < numTerms;
        ++i) { float coefficient;
        int exponent;
        printf("Enter coefficient and exponent for term %d: ", i
        + 1); scanf("%f %d", &coefficient, &exponent);
        insertTerm(&header, coefficient, exponent);
```

```
    }
    return header;
}

void Pwrite(Node* header) {
```

```
    Node* current =
    header; if (current
    == NULL) {
        printf("Polynomial is
        empty.\n"); return;
    }

    do {
        printf("%.2fx^%d ", current->coefficient, current-
        >exponent); current = current->next;
        if (current != header) {
            printf("+ ");
        }
    } while (current != header);
    printf("\n");
}

Node* Padd(Node* a, Node* b)
    { Node* result = NULL;
    Node* aCurrent = a;
    Node* bCurrent = b;

    do {
        insertTerm(&result, aCurrent->coefficient, aCurrent->exponent);
        aCurrent = aCurrent->next;
    } while (aCurrent != a);

    do {
        insertTerm(&result, bCurrent->coefficient, bCurrent->exponent);
        bCurrent = bCurrent->next;
```

```
} while (bCurrent != b);
```

```
   return result;
}

Node* Psub(Node* a, Node* b)
   { Node* result = NULL;
   Node* aCurrent = a;
   Node* bCurrent = b;

   do {
      insertTerm(&result, aCurrent->coefficient, aCurrent->exponent);
      aCurrent = aCurrent->next;
   } while (aCurrent != a);

   do {
      insertTerm(&result, -bCurrent->coefficient, bCurrent-
      >exponent); bCurrent = bCurrent->next;
   } while (bCurrent != b);

   return result;
}

Node* Pmult(Node* a, Node* b)
   { Node* result = NULL;
   Node* aCurrent = a;

   do {
      Node* bCurrent =
      b; do {
         insertTerm(&result, aCurrent->coefficient * bCurrent->coefficient, aCurrent-
>exponent + bCurrent->exponent);
         bCurrent = bCurrent->next;
```

```
    } while (bCurrent !=
    b); aCurrent =
    aCurrent->next;
  } while (aCurrent != a);

  return result;
}

float Peval(Node* header, float
  a) { float result = 0.0;
  Node* current = header;

  do {
    result += current->coefficient * pow(a, current-
    >exponent); current = current->next;
  } while (current != header);

  return result;
}

void Pearse(Node** header, int
  exponent) { if (*header == NULL) {
    printf("Polynomial is
    empty.\n"); return;
  }

  Node* current = *header;
  Node* previous =
  NULL; Node* tail =
  *header;
```

```
do {
   if (current->exponent == exponent) {
```

```
        if (previous == NULL) {
          *header = current-
          >next; tail->next =
          *header;
          free(current);
          printf("Term with exponent %d erased.\n",
          exponent); return;
        } else {
          previous->next = current->next;
          free(current);
          printf("Term with exponent %d erased.\n",
          exponent); return;
        }
      }
      previous = current;
      current = current-
      >next;
    } while (current != *header);

    printf("Term with exponent %d not found.\n", exponent);
}

void freeList(Node**
    header) { if (*header ==
    NULL) {
      return;
    }

    Node* current = *header;
    Node* nextNode = current->next;
```

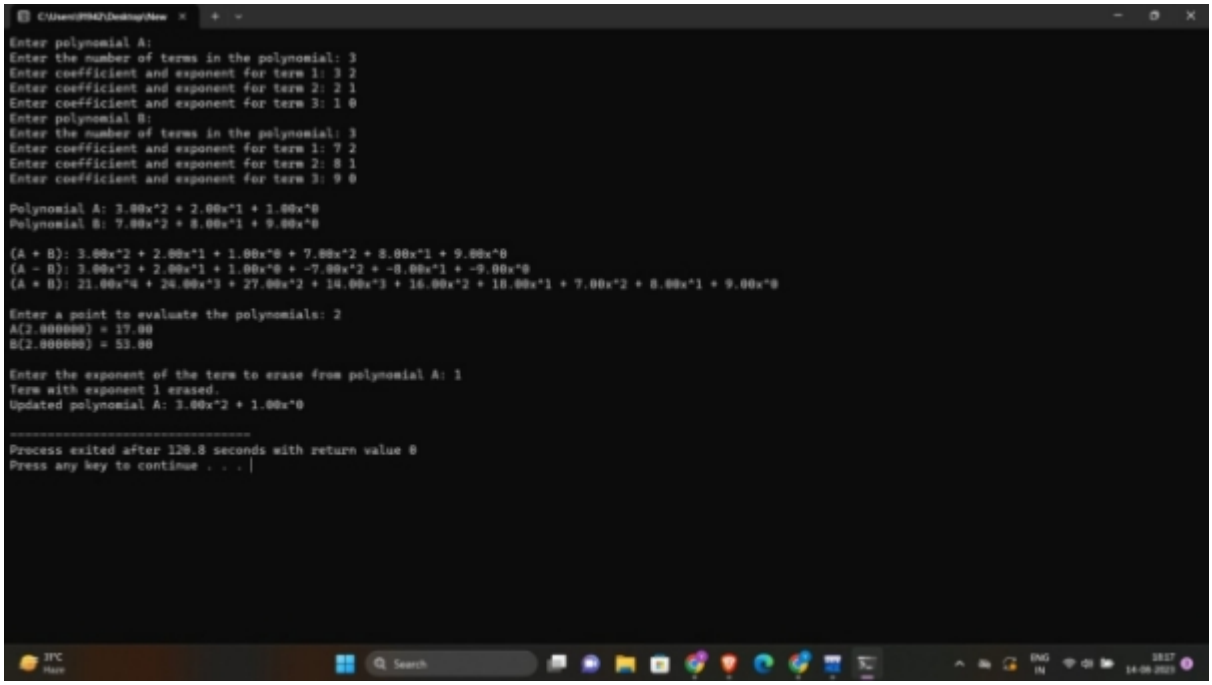
```
do {
   free(current);
   current =
   nextNode;
```

```
 nextNode = current->next;
        } while (current !=
                 *header);

   *header = NULL;
}

int main() {
   Node*  polyA  =
   NULL;    Node*
   polyB  =  NULL;
   Node*  result  =
   NULL;       float
   evalPoint;

   printf("Enter polynomial
   A:\n"); polyA = Pread();
   printf("Enter polynomial
   B:\n"); polyB = Pread();

   printf("\nPolynomial A:
   "); Pwrite(polyA);
   printf("Polynomial B: ");
   Pwrite(polyB);

   result = Padd(polyA,
   polyB); printf("\n(A +
   B): "); Pwrite(result);
   freeList(&result);
```

```
result = Psub(polyA,
polyB); printf("(A - B):
"); Pwrite(result);
freeList(&result);
```

```
    result = Pmult(polyA,
    polyB); printf("(A * B):
    "); Pwrite(result);
    freeList(&result);

    printf("\nEnter a point to evaluate the
    polynomials: "); scanf("%f", &evalPoint);
    printf("A(%f) = %.2f\n", evalPoint, Peval(polyA,
    evalPoint)); printf("B(%f) = %.2f\n", evalPoint,
    Peval(polyB, evalPoint));

    int eraseExponent;
    printf("\nEnter the exponent of the term to erase from polynomial
    A: "); scanf("%d", &eraseExponent);
    Pearse(&polyA, eraseExponent);
    printf("Updated polynomial A: ");
    Pwrite(polyA);
    freeList(&polyA);

    freeList(&polyB);

    return 0;
}
```

OUTPUT:-


Enter polynomial A:
Enter the number of terms in the polynomial: 3
Enter coefficient and exponent for term 1: 3 2
Enter coefficient and exponent for term 2: 2 1
Enter coefficient and exponent for term 3: 1 0
Enter polynomial B:
Enter the number of terms in the polynomial: 3
Enter coefficient and exponent for term 1: 7 2
Enter coefficient and exponent for term 2: 8 1
Enter coefficient and exponent for term 3: 9 0
Polynomial A: 3.00x^2 + 2.00x^1 + 1.00x^0
Polynomial B: 7.00x^2 + 8.00x^1 + 9.00x^0
(A + B): 3.00x^2 + 2.00x^1 + 1.00x^0 + 7.00x^2 + 8.00x^1 + 9.00x^0
(A - B): 3.00x^2 + 2.00x^1 + 1.00x^0 + -7.00x^2 + -8.00x^1 + -9.00x^0
(A * B): 21.00x^4 + 24.00x^3 + 27.00x^2 + 14.00x^3 + 16.00x^2 + 18.00x^1 + 7.00x^2 + 8.00x^1 + 9.00x^0
Enter a point to evaluate the polynomials: 2
A(2.000000) = 17.00
B(2.000000) = 53.00
Enter the exponent of the term to erase from polynomial A: 1
Term with exponent 1 erased.
Updated polynomial A: 3.00x^2 + 1.00x^0
Process exited after 120.8 seconds with return value 0
Press any key to continue . . .
Search