|  |  |
| --- | --- |
| **Name** | **Hatim Yusuf Sawai** |
| **UID no.** | **2021300108** |
| **Experiment No.** | 4 |

|  |  |
| --- | --- |
| **AIM:** | Apply the concept of recursion to solve a given problem. |
| **Program 1** | |
| **PROBLEM STATEMENT:** | Write a recursive function to find the factorial of a number and test it. |
| **ALGORITHM:** | 1. START (fact) 2. If n==0   Return 1   1. Return n\*fact(n-1) 2. START (main) 3. Input n 4. Output fact(n) 5. STOP |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include<stdio.h>  int fact(int);  int main()  {      int n;      printf("Enter a number:\n");      scanf("%d",&n);      printf("%d! = %d",n,fact(n));      return 0;  }  int fact(int n)  {      if(n==0)          return 1;      return n\*fact(n-1);  } |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT:** | Write a recursive function which returns the nth term of the fibonacci series. Call it from main() to find the 1st n numbers of the fibonacci series. |
| **ALGORITHM:** | 1. START 2. Input n 3. For i=0 4. Output fib(i) 5. Repeat step 4 till i<n 6. STOP 7. START (fib) 8. If n==0 9. return 0 10. else if n==1 11. return 1 12. else return fib(n-1)+fib(n-2); |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include<stdio.h>  int fib(int n);  int main()  {      int n,i;      printf("Enter the range of the series:\n");      scanf("%d",&n);      for(i=0;i<n;i++)      {          printf("%d ",fib(i));      }      return 0;  }  int fib(int n)  {      if(n==0)          return 0;      else if(n==1)          return 1;      else return fib(n-1)+fib(n-2);  } |
| **RESULT:** | |
| **Program 3** | |
| **PROBLEM STATEMENT:** | Given a number n, print following a pattern without using any loop. Example: Input: n = 16 Output: 16, 11, 6, 1, -4, 1, 6, 11, 16 Input: n = 10 Output: 10, 5, 0, 5, 10 |
| **ALGORITHM:** | 1. START 2. Intput n 3. Series(n) 4. STOP 5. START (series) 6. If n<=0 7. Output n 8. return 0; 9. Output n 10. series(n-5) 11. Output n |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include<stdio.h>  int series(int);  int main()  {      int n;      printf("Enter a number:\n");      scanf("%d",&n);      series(n);      return 0;  }  int series(int n)  {      if(n<=0)      {          printf("%d ",n);          return 0;      }      printf("%d ",n);      series(n-5);      printf("%d ",n);  } |
| **RESULT:** | |
| **Program 4** | |
| **PROBLEM STATEMENT:** | Ackerman’s function is defined by:  A(m,n) = n+1 if m=0  = A(m-1,1) if m≠0 and n=0  =A(m-1, A(m,n-1)) if m≠0 and n≠0  Write a function which given m and n returns A(m,n). Tabulate the values of A(m,n) for all m in the range 1 to 3 and all n in the range 1 to 6. |
| **ALGORITHM:** | 1. START 2. Input m, n 3. For i=1 4. For j=1 5. Output ack(j, i) 6. Repeat step 5 till j<=m 7. Repeat Steps 4-6 till i<=n 8. STOP 9. START (ack) 10. If m==0 11. Return n+1 12. Else if n==0 && m!=0 13. return ack(m-1,1) 14. Else if m!=0 && n!=0 15. return ack(m-1,ack(m,n-1)) |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include<stdio.h>  int ack(int,int);  int main()  {      int n, m, i, j;      printf("Enter range(m,n)\n");      scanf("%d %d", &m, &n);      printf("Ackermann series:\n");      printf("m,n     m=1         m=2         m=3\n");      for (i = 1; i <= n; i++)      {          if(i>=10)              printf("n=%d   ", i);          else              printf("n=%d    ", i);          for (j = 1; j <= m; j++)          {              if(ack(j,i)>=10 && i<10)                  printf("A(%d,%d)=%d    ", j, i, ack(j, i));              else if(ack(j, i)>=10 && i>=10)                  printf("A(%d,%d)=%d   ", j, i, ack(j, i));              else                  printf("A(%d,%d)=%d     ", j, i, ack(j, i));          }          printf("\n");      }  return 0;  }  int ack(int m,int n)  {      if(m==0)          return n+1;      else if(n==0 && m!=0)          return ack(m-1,1);      else if(m!=0 && n!=0)          return ack(m-1,ack(m,n-1));  } |
| **RESULT:** | |
| **Program 5** | |
| **PROBLEM STATEMENT:** | Write a recursive function to return the minimum number of coins of given set of coin values that are required to produce a given amount. For example if you are given set of values {1,4,5}(indicating you had a supply of 1-cent,4-cent and 5-cent coins), and the amount 8, you should return 2, since the value 8 cents can be made with two 4-cent coins. |
| **ALGORITHM:** | 1. START 2. Input n 3. For i=0 4. Input c[i] 5. Repeat step 4 till i<n 6. Input amt 7. Output mincoins(c, n, amt) 8. STOP 9. START (mincoins) 10. If amt==0 11. Return 0 12. int res = \_\_INT\_MAX\_\_ 13. for i=0 14. sub=0 15. if amt>=c[i] 16. sub = 1 + mincoins(c, n, amt - c[i]) 17. res = min(res, sub); 18. Repeat steps 6-9 till i<n 19. If res == \_\_INT\_MAX\_\_ 20. Return -1 21. Return res |
| **FLOWCHART:** |  |
| **PROGRAM:** | #include<stdio.h>  #include<math.h>  #define min(X, Y) (((X) < (Y)) ? (X) : (Y));  int mincoins(int \*,int,int);  int main()  {      int c[10],amt,n,min;      printf("Enter number of set values:\n");      scanf("%d",&n);      printf("Enter %d set values:\n",n);      for(int i=0;i<n;i++)          scanf("%d",&c[i]);      printf("Enter the amount:\n");      scanf("%d",&amt);      printf("Minimum number of coins required: %d",mincoins(c,n,amt));      return 0;  }  int mincoins(int c*[]*, int n,int amt)  {      if (amt == 0)          return 0;      int res = \_\_INT\_MAX\_\_;      for (int i = 0; i < n; i++)      {          int sub=0;          if (amt >= c[i])          {              sub = 1 + mincoins(c,n, amt - c[i]);              res = min(res, sub);          }      }      if (res == \_\_INT\_MAX\_\_)          return -1;      return res;  } |
| **RESULT:** | |
| **CONCLUSION:** | In this experiment, we learnt the concept of recursion and how to apply recursion and make a recursive function to solve a given C program. |